PICKUP INSPECTOR

Sistema d'informació i mesura de pastilles de guitarra

Autor: Aitor Terradellas Bou
Data: 18/03/2015
Director: Marc Alier
Departament: Enginyeria del Software i Sistemes d'Informació (ESSI)
Titulació: Enginyeria Superior en Informàtica
Centre: Facultat d'Informàtica de Barcelona (FIB)
Universitat: Universitat Politècnica de Catalunya (UPC)

Índex

1 Introducció

Dins el món de les guitarres elèctriques, els mètodes per valorar qualitativament el so de les pastilles son pràcticament nuls. Aquest document recull l'estudi, investigació i desenvolupament d'un analitzador de pastilles de guitarra elèctrica.

1.1. Descripció del problema i situació

Un dels principals elements que defineixen el so de les guitarres i baixos elèctrics és la **pastilla**. Una pastilla és un element electromagnètic format per una bobina de coure i un iman que recull el so de les cordes de la guitarra un cop es toquen. Aquest so es traspassa al cablejat de la guitarra i, acaba a l'amplificador, on se'n deriva el so final. La pastilla és, per tant, l'element original, pel que el so final serà una modificació del captat per la pastilla. Però les pastilles no son úniques i cada una capta el so de manera diferent.

Imatge 1.1. Exemple de pastilla *Seymour Duncan* de tipus *humbucker*

En el moment en que un guitarrista ha de decidir quina pastilla posar-se, es troba amb una gran gama d'elles però molt poc contingut per a valorar-les i comparar-les entre sí. Sovint, el valor que se li dona a una pastilla està representat pel màrqueting, per valors subjectius.

És habitual que les descripcions de les pastilles vinguin donades per adjectius ambigus i poc clars com 'potent' o 'dur', sense entrar massa en detall. També és habitual que el valor que se li pugui donar a aquests components vingui donat pel d'una autoritat musical com pot ser un guitarrista professional. Això podria indicar que aquella pastilla ha estat provada o usada per tal guitarrista i que la recomana, però res ens assegura que això sigui així, doncs l'empresa que ven la pastilla pot haver pagat perquè surti la recomanació del guitarrista en qüestió. De totes maneres, ens trobem a les mateixes, res ens indica les propietats sonores del component.

En pocs casos, alguns valors mesurables s'ofereixen de cara al públic, però en cap cas son valors significatius o no poden donar pas a una valoració. Per exemple, el valor més usat pels comerciants sol

ser la resistència electromagnètica de la pastilla, però aquest valor influeix bastant poc en el so final i, en qualsevol cas, no se'ns defineix quina relació existeix entre el so i el valor de la resistència. Quin so estem obtenint amb una resistència major o inferior? Si la resistència no és important, quins valors ho son?

1.2. Motivació

Com a entusiastes de les guitarres, ens interessa poder trobar aquests mecanismes de mesura i valoració, ja sigui per realitzar unes compres més intel·ligents d'unes pastilles o bé per obtenir el so que més s'adapti al nostre gust musical.

A la vegada, aquest entusiasme es tradueix a una necessitat real, pel que ens converteix en usuaris potencials del producte. Això ens permetrà aportar-li qualitat, usabilitat i *feedback.* Nosaltres som els nostres propis clients, en part.

De cara a la tecnologia, tenim l'oportunitat d'investigar i aprendre sobre diversos camps. Haurem d'aprendre sobre teoria electromagnètica, analitzar i escollir els dispositius hardware que més ens convinguin i explorar noves tecnologies software.

A nivell personal, m'interessa poder realitzar un projecte en un entorn web, doncs és un tema poc tocat a la carrera i que m'agradaria indagar. També resulta una interessant sortida professional i una branca de la informàtica cada dia més important.

Donat a que no existeix un antecedent del projecte, les possibilitats i alternatives son enormes, pel que la pròpia definició de l'arquitectura sigui un repte en si mateixa.

Imatge 1.2. Exemple d'anunci publicitari per pastilles

1.2.1. Aplicacions

A part de les motivacions personals, aquest projecte pot tenir les següents aplicacions.

Comparador de diferents sons

La construcció d'un sistema d'aquest tipus ens pot ajudar a fer una comparació entre diferents pastilles o configuracions d'un circuit electrònic (si incloem altres elements electromagnètics) per tal de buscar un so concret. Així també, podem elegir els components que més s'adaptin a les nostres necessitats.

Fabricació en sèrie

El sistema també pot servir per la fabricació en sèrie. Un fabricant de pastilles li interessa saber si una tirada de pastilles té un so homogeni a cada copia. El sistema li pot permetre veure la similitud o diferència entre aquestes.

D'aquestes dues aplicacions se'n pot deduir una tercera. Si el fabricant decideix incloure aquests dades al seu producte i els consumidors n'estan informats, aquests tindran una major capacitat per valorar diferents models. Això deriva a una compra més conscient.

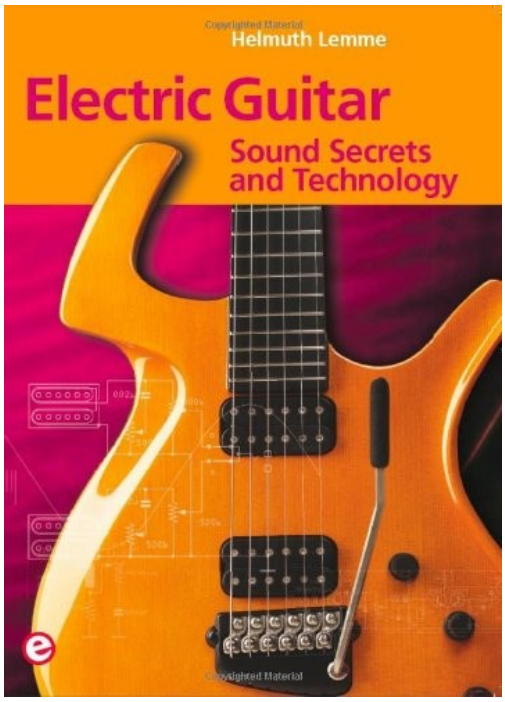
1.3. Objectius

Els objectius del nostre projecte són:

- **Estudi de la teoria**. Per resoldre el nostre problema haurem d'estudiar la teoria necessària i saber exactament els resultats que ens seran útils i aquells que no.
- **Investigació de possibilitats i alternatives.** També ens caldrà buscar i analitzar els dispositius, algoritmes i programes que ens permetin implementar el projecte de la millor manera.
- **Implementació de les mesures.** Un cop tinguem l'arquitectura ens farà aplicar la teoria i certs algoritmes per l'obtenció de les dades amb les que puguem valorar les pastilles.
- **Implementació del sistema d'informació.** A banda d'obtenir les dades, aquestes hauran de ser recollides i emmagatzemades, les haurem de poder mostrar a l'usuari i extreure'n nova informació.

1.4. Base teòrica mínima

Per tal de treballar sobre aquest sistema, fa falta aportar una sèrie de notes sobre la teoria en la que ens basem.
La principal font és *Electric Guitar: Sound Secret & Technology* de Helmuth Lemme. En aquest llibre, Helmuth Lemme ens explica en detall tots els components que formen una guitarra. Això, òbviament, inclou la part electrònica, que és la què ens interessa.



Imatge 1.3. *Electric Guitar: Sounds Secrets & Technology* de Helmuth Lemme

1.4.1. Valors d'entrada

Dins el circuit electrònic, s'hi troba la pastilla, que pot ser tractada com un circuit elèctric a part. Així, la pastilla tindrà diversos valors que la caracteritzen:

- Resistència en sèrie (Rs), mesurada en Ohms (Ω).
- Resistència en paral·lel (Rp), en Ohms (Ω).
- Inductància (L), mesurada en Henries (H).
- Capacitat (C), mesurada en Faradays (F).
- El Volum (Vol), mesurat en Volts (V).
- Voltatge entrant, mesurat en Volts (V).

La resta de components electrònics (potenciòmetres, condensadors, cablejat...) actuen sobre el circuit elèctric de la pastilla com s'hi fossin un filtre de pas baix. Aquest filtre té les següents valors:

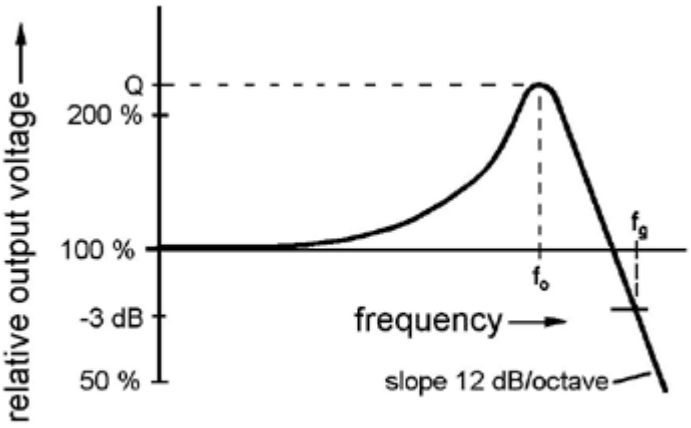
- Resistència (R), mesurada en Ohms (Ω).
- Capacitat (C), mesurada en Faradays (F).

1.4.2. Valors de sortida

A partir d'aquest valors entrants, podem obtenir els valors de sortida, aquells que ens descriuran el so final del nostre instrument. Aquests valors son:

- La freqüència de ressonància (F o F0), mesurada en Hertzs (Hz).
- La qualitat (Q), mesura sense unitats.

Aquestes característiques s'obtenen a partir d'un escombrat del rang de freqüències audibles (de 0Hz a 20kHz) sobre una funció de transferència pròpia del circuit. Depenent dels valors d'entrada, aquest escombrat generarà una resposta a cada una de les freqüències tractades, creant una corba. Donat el filtre de pas baix, existirà un punt a la corba on se'ns mostrarà un pic de ressonància i, poc més endavant, la corba decaurà a 20 dB/dècada. Així doncs, tots els harmònics a freqüències majors seran atenuats, els harmònics a freqüències properes al pic seran amplificats i la resta, no seran modificats. La freqüència on es troba aquest pic és la que anomenarem freqüència de ressonància. La relació que existeix entre el voltatge d'entrada i la resposta en aquest punt (és a dir, l'altura del pic de ressonància) és el valor de la qualitat.



Imatge 1.4. Gràfic d'una funció de transferència

1.4.3. Com afecten els valors de sortida

Si considerem *f* com la freqüència de ressonància d'un circuit, la classificació és la següent:

- si *f* ronda els 1500Hz: el so és molt **suau** (soft), **apagat** (dull).
- si *f* no passa de 2000Hz: so **greu** (fat), propi del **blues** (bluesy).
- si *f* no passa de 2500Hz: conté un so **intermedi**, ni apagat ni estrident (warm, full).
- si *f* no passa de 3000Hz: el so és **brillant** (brilliant).
- si *f* no passa de 4000Hz: típic so Fender, **metàl·lic**, **agut** o **twangy** (metal·lic hard).
- si *f* volta els 5000Hz-6000Hz: so **accentuat** o **escarpat**, molt **agut** (piercing).
- si *f* està per sobre dels 7000Hz: so **trencadís** i **fràgil** (brittle, glassy), propi de les guitarres acústiques.

Per tant, quant més alta sigui la freqüència de ressonància, més freqüències agudes deixarà passar el filtre i més agut serà el so de la nostra guitarra.

En relació a la qualitat, s'ha de dir que no és una dada que pugui ser tan classificable com la freqüència de ressonància. El que ens dóna la qualitat és l'expressivitat de la freqüència de ressonància. Com ja s'ha dit, els harmònics propers al pic de ressonància estaran amplificats, mentre que els que estiguin en freqüències majors seran atenuats dràsticament. Aquesta amplificació i atenuació estaran més pronunciats quant més alta sigui la qualitat de la nostra mesura.

1.5. Antecedents

Donat a que els valors de la pastilla no ens són donats pel fabricant, farà falta mesurar-los, així com mesurar els dos valors de sortida.

Fins ara, l'única teoria remarcable que hem trobat ha sigut el llibre *Electric Guitar: Secret Sound & Technology* de Helmuth Lemme que explica detalladament la teoria electromagnètica necessària, els mètodes d'extracció de dades i algunes referències sobre la valoració final.

En el llibre, Helmuth presenta una primera eina per mesurar les característiques de les pastilles: *Pickup Analyzer*. És un dispositiu electrònic que ens permet realitzar aquestes mesures a través de diverses configuracions i guardar els resultats, representats en una gràfica, a través d'un petit software, al nostre PC.

Imatge 1.5. *Pickup Analyzer*

Aquesta eina però, es queda curta en diversos aspectes:

a. El valor final de la guitarra ve donat per diversos factors, no només la pastilla. *Pickup Analyzer* dona certes opcions de mesura però fa falta molta més informació a tenir en compte.

b. No existeix un sistema d'informació que li doni suport. La segona versió de l'analitzador permet connectar-lo al PC però el software que conté és molt reduït, sent només possible obtenir algunes gràfiques.

c. *Pickup Analyzer* és un producte comercial que no permet modificar el seu funcionament, pel que és més propens a tenir errors i a no poder arreglar-los.

d. L'analitzador funciona de manera manual, pel que seria interessant poder automatitzar el mètode de mesura per agilitzar un procés d'obtenció de dades.

e. *Pickup Analyzer* només ens permet obtenir el valor de la freqüència de ressonància i la qualitat, però no passem per un pas intermedi on s'obtenen la resta de valors de la pastilla.

Com ja s'ha comentat, partim de 0 en la creació del sistema d'informació, doncs no existeix un software que satisfaci els nostres objectius.

Per tant, el que ens tocarà fer serà crear un sistema electrònic per mesurar els valors de la pastilla, un sistema per a mesurar els valors de sortida i un sistema d'informació que reculli aquests valors i els presenti.

1.6. L'usuari

Donat a que fem aquest projecte per motivació personal i no per encàrrec, el nostre usuari final serà qualsevol fanàtic de les guitarres que busqui millorar el so de la seva guitarra o algun comerciant que vulgui millorar la qualitat dels seus productes.

L'usuari final, per tant, pot respondre a molts perfils diferents. Serà important poder arribar tant a gent que pot entendre el funcionament elèctric d'una pastilla com a guitarristes principiants. Farà falta aportar dades a l'usuari que aquest pugui entendre i valorar sense massa problema.

També s'haurà de tenir en compte l'accessibilitat dels usuaris, doncs el projecte pretén ser una eina a l'abast popular. Això també significa que hem de cuidar el tema econòmic i reduir el preu del producte el màxim possible.

1.7. Anàlisi d'impactes i beneficis

1.7.1 En l'organització i els seus usuaris

Donat a que el projecte parteix d'una idea nova, no genera un canvi als usuaris, doncs aquests també son nous. L'usuari no perd res, sinó que hi guanya un nou mètode per valorar el seu circuit. L'impacte és el propi sistema.

D'altra banda, el sistema està basat en *Arduino* i codi obert, pel que permet una font de coneixement i millora del sistema molt accessible per l'usuari. Aquest guanya transparència amb el sistema.

1.7.2 Economia

El cost del sistema equival al cost de l'*Arduino* i els seus complements, doncs fins el moment el servidor és gratuït i l'aplicació web està servida pel servidor.

1.7.3 Riscos

Els dos riscos principals son la usabilitat i l'escalabilitat. El sistema pretén ser intuïtiu per a l'usuari, però noves eines com *Arduino* i un poc coneixement sobre electrònica poden causar rebuig dels usuaris més novells. L'aplicació web no conté riscos degut a la seva senzillesa.

D'altra banda, els resultats generats per les mesures poden ser confusos per l'usuari si no entén els conceptes claus. De cara a l'escalabilitat, el sistema no està pensat per la interacció exhaustiva, pel que pot portar a errors i confusions. Això pot afectar molt negativament en cas de males pràctiques per part dels usuaris.

Un altre risc és la precisió. El sistema calcula els valors dinàmics a través d'un algoritme programat. Aquest algoritme té les seves limitacions, doncs contempla casos molt concrets i l'ús d'altres eines més avançades pot donar resultats més satisfactoris.

Per últim, el sistema està programat sobre *Python* al servidor i alguns dels algoritmes s'executen en *Javascript* al mateix client. Son llenguatges interpretats, pel que la velocitat de reacció de les interfícies pot reduir-se en alguns casos.

2 Subsistemes i funcionalitats

2.1. Visió general i arquitectura

Per a crear aquest sistema, separarem les diverses funcions que haurà de tenir en diferents subsistemes.

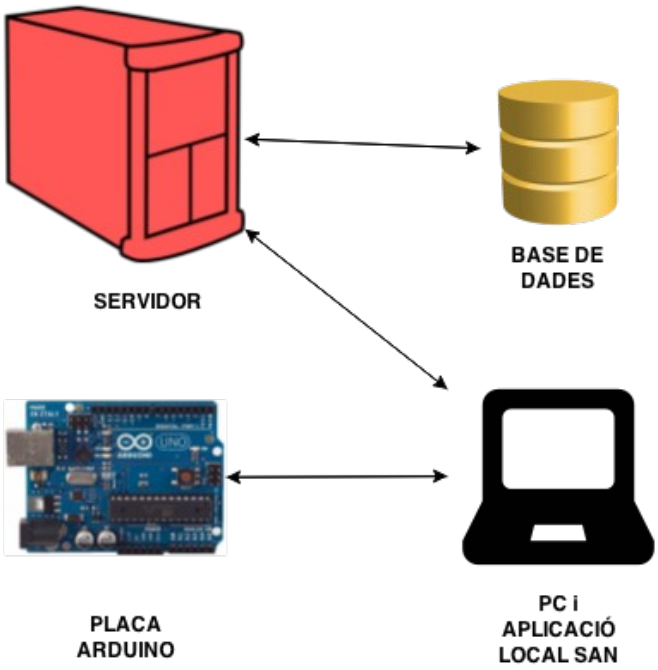
- Per una banda tindrem el **SAN** (Subsistema d'Anàlisi). Aquest subsistema tractarà tot el tema de les mesures dels valors, tant els de les pastilla com els valors de sortida.
- En segon lloc tindrem el subsistema de disseny de circuits **SED** (Subsistema d'edició i disseny). En aquesta part s'implementarà la capacitat per representar els nostres circuits al sistema, de manera que existeixi una relació entre els valors mesurats i la representació física d'aquest.
- També farà falta que aquest valors siguin guardats en algun lloc i hi hagi una comunicació entre els subsistemes. És per això que necessitarem **SREC** (Sistema de recol·lecció).
- Per últim, voldrem poder presentar les dades recollides i modificar-les. Per tant, ens farà falta un sistema d'informes o *reporting* **SREP**.

Tot i la separació feta, es pretén integrar els subsistemes entre ells de manera que es comuniquin de manera semblant i es trobin dins una plataforma comuna. L'arquitectura final queda de la següent manera:

Per una banda, tindrem el servidor, que serà l'encarregat de la persistència de les dades i del proveïment de les funcionalitats de SED, SREP i algunes de SAN. L'usuari podrà accedir a aquestes funcionalitats a través d'una interfície web pública.

Tindrem un programa local encarregat d'alguns funcionalitats de SAN i la comunicació amb el dispositiu físic de mesura, que estarà implementat sobre una placa d'*Arduino*.

Totes les comunicacions entre plataformes formen part del subsistema SREC.



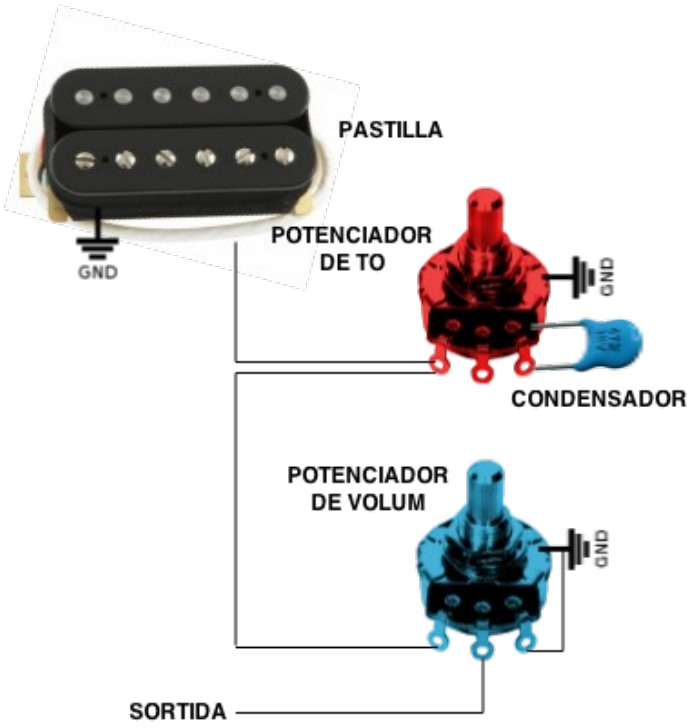
Imatge 2.1. Arquitectura del sistema

2.2. Patrons de disseny

Com explicarem als següents apartats, la implementació d'un sistema que tingui en compte totes les configuracions possibles per pastilles i altres elements és bastant complexa. El nostre sistema només tindrà en compte una sèrie de paràmetres que hem sabut caracteritzar i una configuració d'un circuit concret.

Es tractarà d'un circuit amb una pastilla, un potenciòmetre de volum i un altre de to. Aquests elements estan connectats en sèrie (la pastilla es connecta al potenciòmetre de to, aquest, al de volum i aquest es connecta a la sortida de la guitarra), pel que anomenarem aquest circuit **serial_1_2** (en sèrie, 1 pastilla i 2 potenciòmetres).

No obstant, la nostra idea és que el projecte pugui tenir una continuïtat després d'haver fet la presentació, pel que ens interessa deixar-lo preparat perquè es pugui reutilitzar la major part del codi per a altres configuracions de circuits i poder implementar les diferències entre configuracions sense errors en el que ja tenim. És per això que aplicarem el patró de disseny *Layout*.



Imatge 2.2. Esquema d'un circuit *serial_1_2*

Aquest consistirà en estendre la classe *Circuit* a noves subclasses que representin les diferents configuracions. A cada operació sobre un circuit haurem de valorar si la funció pot ser generalitzada i pot servir per qualsevol instància o bé fa falta implementar-la a part per cada configuració. Aquestes decisions les veurem reflectides a cada subsistema explicat a continuació.

Donades les decisions sobre les classes usades i els seus atributs, aquest patró de disseny també s'haurà d'aplicar sobre la classe *Measure* (Mesura), que heretarà el tipus de subclasse del circuit que estiguem mesurant.

2.3. Lògica del sistema

Donat a que el sistema parteix d'una idea nova, pot arribar a ser confús la lògica de funcionament i el flux d'informació i per això creiem necessari descriure'l.

L'objectiu final del programa és poder tenir a la nostra disposició la informació sobre les mesures fetes sobre un circuit. Per arribar a fer això, primer haurem de passar per una fase de creació i disseny del nostre circuit, que serà implementada dins el subsistema SED. A aquest circuit però, li faran falta els valors dels paràmetres de la pastilla, pel que farà falta fer la mesura d'aquests paràmetres al subsistema SAN. Per últim es realitzarà la mesura de la freqüència de ressonància i la qualitat que també forma part del subsistema SAN. Per últim, aquestes dades podran ser consultades i gestionades al subsistema SREP. Per la permanència de les dades i la comunicació entre subsistemes serà feina del subsistema SREC.

Imatge 2.3. Gràfic de flux de la lògica del sistema

2.4. Sistema d'anàlisi (SAN)

2.4.1. Introducció

Com ja hem comentat, SAN serà el sistema encarregat de les mesures dels valors de les pastilles i els valors de sortida.

2.4.2. Mesures estàtiques i mesures dinàmiques

Abans de parlar sobre SAN, fa falta aclarir la distinció entre les mesures estàtiques i les dinàmiques. Això fa referència als mètodes de mesura que farem servir sobre els diferents valors.

Per una banda, tindrem els valors estàtics. Aquests valors son els que s'obtenen de la pastilla, és a dir, les resistències, la capacitat, el voltatge entrant i la inductància. Donat a que aquests valors son únics en la pastilla i no varien, els anomenem estàtics.

Els valors dinàmics son els valors de sortida: la freqüència de ressonància i la qualitat. Com ja hem comentat al principi, aquests valors s'obtenen a partir d'un escombrat sobre un rang de freqüències pel que ens farà falta aplicar-hi un algoritme. Per això els anomenarem valors dinàmics.

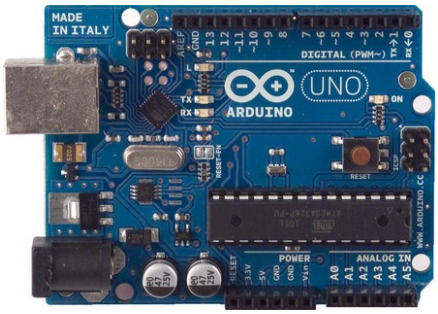
Donat a la seva diferència, els dos tipus de valors es mesuraran en parts diferents del SAN.

2.4.3. Maquinari

Els valors estàtics hauran de ser, necessàriament, calculats analògicament, pel que farà falta una plataforma per passar aquests valors d'analògics a digitals. En primera instància es van proposar diversos dispositius.

Cable USB Rocksmith
Aquest cable és el que s'usa a la sèrie de videojocs *Rocksmith*. Es tracta d'un videojoc basat en el maneig d'una guitarra real que es connecta a la consola. Aquest cable és capaç de transmetre el so de la guitarra a través d'una entrada USB. Aquesta opció va ser destriada al principi del projecte donat a que no es va trobar cap manera de poder controlar l'USB a partir d'un altre programa que no fos *Rocksmith*. De totes maneres, les dades que podríem extreure d'aquest cable series freqüències a analitzar a través de mètodes diferents. Per tant, els resultats obtinguts no ens haguessin servit massa.

Imatge 2.4. Cable *Rocksmith*



Imatge 2.5. Placa d'*Arduino UNO*

Arduino
Es tracta d'una plataforma de prototipatge. A partir d'una placa base molt senzilla i un software distribuït gratuïtament, podem realitzar petits projectes en codi *C*. La placa d'*Arduino* té diversos pins on s'hi poden connectar elements com botons, cablejat o leds o inclús plaques extres d'*Arduino* que ens permeten noves funcionalitats (*Ethernet*, font de corrent, etc). També tenim un port sèrie que és on connectarem l'*Arduino* al nostre PC.

Podem trobar el software a la pàgina oficial d'*Arduino*, en diferents versions pels sistemes operatius més comuns (*Windows*, *Mac OS*, *Linux*). Aquest programari inclou un editor de textos amb una interfície per compilar i pujar el codi que volem fer servir.

Donada la flexibilitat d'ús físic i capacitat de programació, *Arduino* semblava el més adient per a SAN. Gràcies a l'enorme comunitat creada a Internet, podem trobar diversos mètodes per mesurar els valors estàtics. Tot i això, no teníem experiència amb la plataforma.

Durant el mes de febrer, l'associació de robòtica de la UPC **AESS** organitzava un curs d'introducció a *Arduino*, seguit d'un de més avançat. Per aquesta raó, vam decidir utilitzar *Arduino*.

El model de placa d'*Arduino* que hem fet servir ha sigut el que ve dins l'*Starter Kit* (paquet de principiant), *Arduino UNO Rev3*. No s'ha considerat cap altre model degut a que no ens fan falta unes capacitats especials pel treball que hem de realitzar.

2.4.4. Coordinació i divisió del treball

Ja tenim el maquinari triat però, com el farem servir per obtenir els valors que volem? A principis de Febrer, ens arriba la proposta d'un projectant de l'ETSETB (Telecomunicacions) interessat a coordinar-se amb el nostre projecte. En **Javier Vallès** realitza un projecte de cara a documentar la base teòrica que farem servir. Tot i tenir ja una experiència amb *Arduino*, hi haurà mètodes que no sabrem implementar, dubtes i errors a resoldre.

Així doncs i, per qüestions acadèmiques, describim l'abast de SAN que assumirà el nostre projecte i quina part assumirà el projecte d'en Javier. Per la part d'en Javier:

- Definició i espeficicació de les mesures.
- Disseny i muntatge del maquinari (Arduino).
- Implementació del codi de les mesures d'Arduino (mesures estàtiques).

Per nosaltres ens queda:

- Implementació del codi per la web (mesures dinàmiques).
- Comunicació entre Arduino i la resta del sistema.

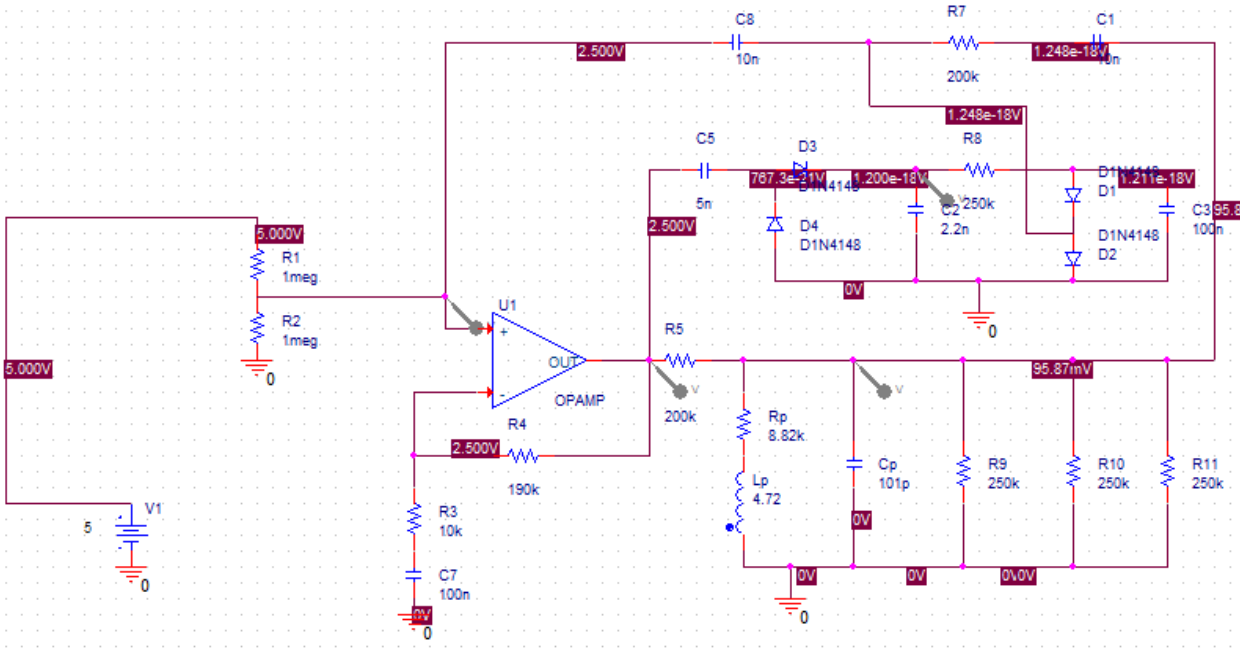
2.4.5. Algorítmia

Per fer les mesures dels valors estàtics usarem *Arduino* però, decidim usar una alternativa per mesurar els valors dinàmics. Aquesta serà la implementació de la mesura dels valors dinàmics a través d'un algoritme programat. Així doncs, un cop hem obtingut els valors estàtics, ja no farà falta l'ús d'*Arduino*. Per una banda, aquesta decisió ens permet avançar feina en paral·lel: la implementació de la mesura dels valors estàtics és una feina molt centrada en el treball d'en Javi, pel que nosaltres podem començar a treballar sobre altres aspectes del sistema sense necessitat de tenir aquesta part acabada. D'altra banda eliminem complexitat a l'*Arduino* i la centrem al PC o servidor, al que hi tenim més control.

En contrapartida, depenent de la plataforma que usem per fer aquestes mesures, la complexitat, eficiència i precisió recaurà sobre aquesta.

L'opció de mesurar els valors dinàmics a través de l'*Arduino* hagués sigut el que usa *Pickup Analyzer*: a partir d'un generador d'ones sinusoïdals, s'alimenta una bobina transmissora. Aquesta s'acosta a la pastilla de la guitarra i es mesura el voltatge de sortida d'aquesta amb un multímetre o oscil·loscopi. Aquí és on variem la freqüència de l'ona sinusoïdal, fent l'escombrat de freqüències. La resposta de cada freqüència variada és el que ha de dibuixar la corba que ens indicarà el valor de la freqüència de ressonància i la qualitat. Donat a que no utilitzem aquest mètode, hem deixat a part alguns detalls sobre aquest mètode que son consultables al llibre de Helmuth Lemme o a la pàgina web *buildyourguitar.com*.

Mesura dels valors estàtics



Imatge 2.6. Esquema de l'oscil·lador per trobar les mesures

Pels valors estàtics s'ha utilitzat un oscil·lador implementat per en Javier a partir de l'esquema de la imatge 2.6.

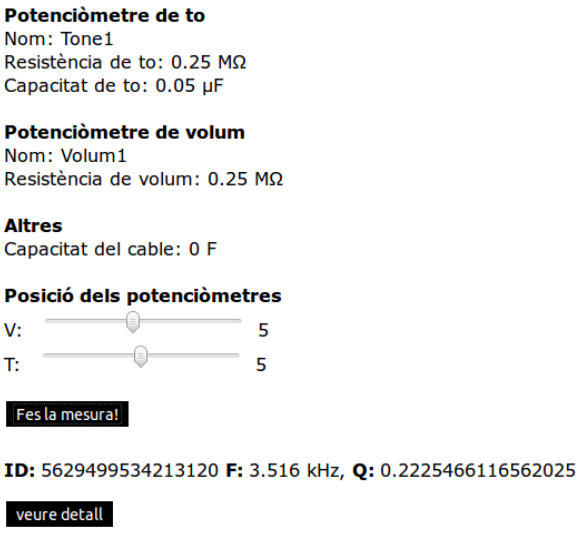
Per fer les mesures es fa oscil·lar el circuit a la freqüència de ressonància de la pastilla i, a partir de condensadors de proba que connectem i desconnectem amb l'*Arduino*, trobem els valors de la inductància, capacitat i les resistències en paral·lel i en sèrie de la pastilla.

Aquests condensador es troben connectats als registres R5 i R7 i son activats per l'Arduino a partir de transistors.

Mesura dels valors dinàmics

Com ja hem comentat, usarem un algoritme programat per obtenir la freqüència de ressonància i la qualitat. Partim dels valors estàtics de la pastilla i valors d'altres components del circuit elèctric.

D'aquí en derivem una funció de transferència. Els paràmetres que defineixen l'equació de la funció de transferència poden ser molts, que vindran donats per la quantitat de components que tingui el nostre circuit. A més components, la funció de transferència serà més complicada, pel que el càlcul de la resposta a cada freqüència serà més pesat. Donat a aquests factors i la falta de més temps per desenvolupar més aquesta equació, ens hem obligat a limitat els paràmetres:



Imatge 2.7. Interfície per les mesures dinàmiques

- Els valors estàtics d'una pastilla, descrits anteriorment.
- Un potenciòmetre de volum que no és més que una resistència mesurada en Ohms i el valor de la posició del potenciòmetre, que serà un valor de 0 a 1.
- Un potenciòmetre de to, també una resistència i un valor de posició. El to també inclou el valor de la capacitat del seu condensador, mesurada en Faradays.
- També s'inclou un valor opcional, que és la capacitat del cable, que ve donat en Faradays.

A part dels valors estàtics de la pastilla, la resta de paràmetres es poden consultar a la fitxa tècnica del producte. Amb aquests paràmetres ja podem construir la funció de transferència, que serà la que ens dibuixarà la corba sobre el rang de freqüències. El rang de freqüències ha de ser audible per l'oïda humana. Segons la teoria, aquest és de 0Hz a 20kHz.

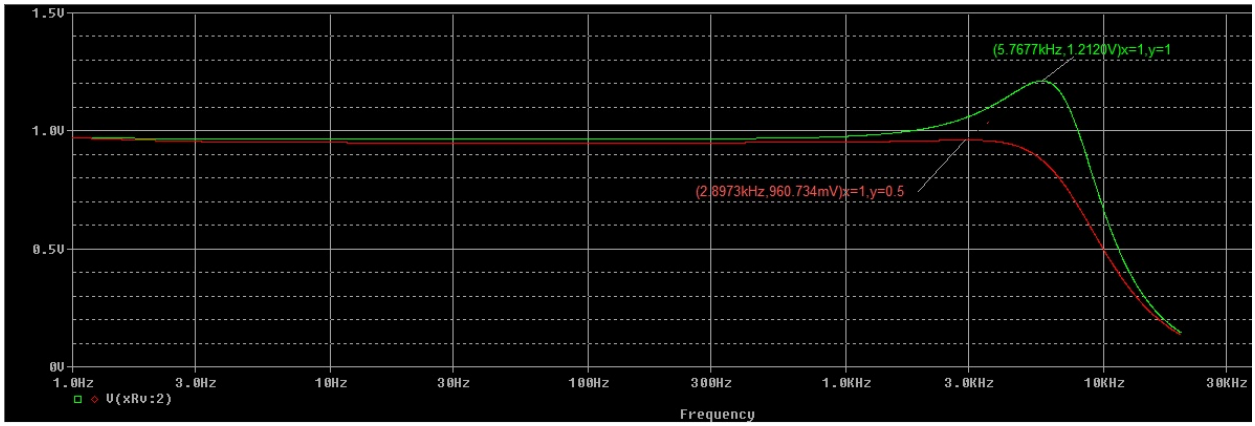
Quan realitzem aquest escombrat, busquem la freqüència on el retard de fase de la senyal és de -90 graus**.** Com que l'algoritme és aproximat, la fase més propera a π/2 (és a dir, la fase més petita) serà la que representa el pic de ressonància. Aquesta freqüència representa la freqüència de ressonància i la seva amplitud, el factor Q.

Aquest algoritme es cridarà des d'una interfície web integrada en tot el sistema. Donat al disseny proposat, per realitzar una mesura dels valors dinàmics haurem de triar un dels circuits que ja continguin els valors estàtics de les pastilles. A cada mesura li haurem d'indicar les posicions dels potenciòmetres de volum i to, que seran valors entre 1 i 10.

Proves

Donat a que fem una feina en paral·lel, la mesura dels valors estàtics ens arriba després d'implementar la mesura dels valors dinàmics. Per tal de conèixer la correctesa de la mesura dels valors dinàmics, partirem d'uns valors estàtics d'exemple. Aquests valors han estat calculats al laboratori d'electrònic del Campus Nord a partir de mesuradors especialitzats i eines de software com *Matlab*.

També s'ha calculat el valor de la qualitat i la freqüència de ressonància a partir d'un oscil·loscopi i eines de disseny de circuits elèctrics tipus *SPICE*. Aquest càlcul ens servirà per assegurar-nos que l'algoritme està realitzat correctament. Els valors obtinguts a SAN no son els mateixos obtinguts en aquestes mesures de prova però s'hi apropen molt. Donat a que el nostre sistema està limitat, és normal que el resultat obtingut no coincideixi al 100%, pel que considerem que els valors de les mesures son correctes.



Imatge 2.8. Dades de laboratori per realitzar les primeres proves de SAN

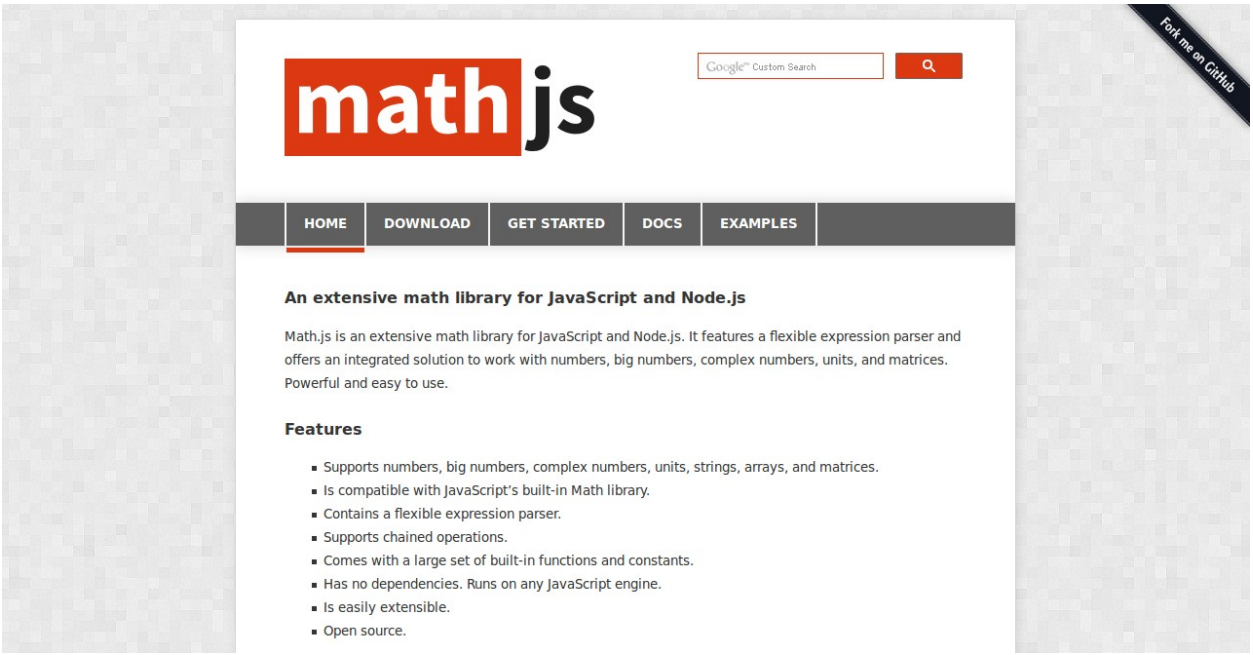
Aquests mesuradors no han sigut contemplats com una opció per a la mesura genèrica dels valors estàtics ni dinàmics, doncs una de les nostres intencions és que el sistema resultant sigui accessible a un usuari final. Tant el preu com la complexitat d'aquestes màquines i programes fan que siguin inaccessibles al públic general.

2.4.6. Software

Càlcul de les mesures dinàmiques

La interfície d'usuari per realitzar el càlcul està disposat com una pàgina *HTML*. En aquesta pàgina triarem el circuit sobre el que fer-hi la mesura i la posició dels potenciòmetres. Un cop fet això, indicarem que volem fer la mesura i, al cap d'uns segons, rebrem el resultat, tot mostrant-nos l'identificador de la mesura (valor auto-generat), freqüència de ressonància i qualitat.

El programari pel càlcul de les mesures dinàmiques ha estat codi *Javascript*. En una primera versió es va incloure la llibreria *math.js* que ens permetia treballar amb nombres complexes necessaris per la funció de transferència.



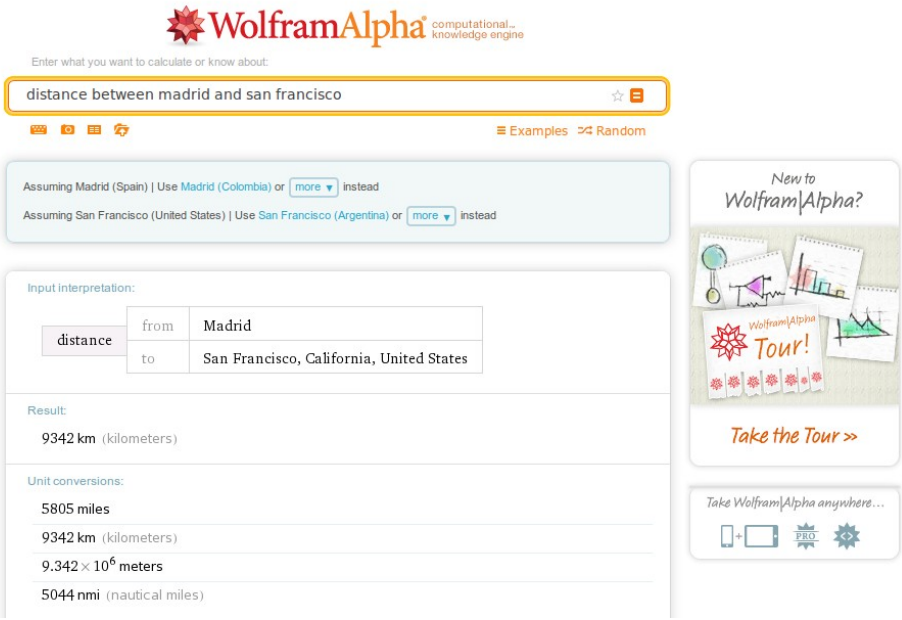
Imatge 2.9. Pàgina d'inici de *math.js*

Reconeixem que *Javascript* no resulta una molt bona opció per a càlcul matemàtic, doncs es tracta de codi interpretat i, depenent de la mida de les dades, la resposta pot tardar un temps a arribar. Tot i això, la facilitat de programació i la fàcil adaptació en entorns web, ens ha fet seguir amb aquest llenguatge.

Per millorar l'eficiència, una proposta va ser reduir la mida de la informació tractada. Fins ara fèiem l'escombrat sobre 20000 valors, corresponents a cada freqüència. Per fer el càlcul més ràpid, es podria tractar una freqüència de cada 10Hz, pel que tractaríem 2000 valors. Es perdria precisió en el càlcul, però a efectes pràctics, el resultat del so no varia en 10Hz.

En una segona versió d'aquest codi, es reconeix que no fa falta la representació mateixa dels nombres complexes, sinó el valor dels coeficients de la part imaginaria i real, per separat. Es per això que s'ha millorat el codi per tal d'evitar l'ús de nombres complexos. A partir d'aquí, ja ens és innecessari l'ús de la llibreria *math.js*. A la vegada, el fet de no usar aquesta llibreria ha fet millorar dràsticament l'eficiència de l'algoritme, tenint la possibilitat d'executar l'escombrat de 20000 valors.

Una de les opcions que s'havien explorat era usar l'aplicació de computació remota **Wolfram Alpha**. Es tracta d'una eina disponible al públic general a través del navegador que ens permet consultar-hi càlculs matemàtics però també realitzar-hi preguntes més comunes i en llenguatge natural (per exemple, podem preguntar-li la distància entre Madrid i San Francisco). Aquesta eina rebria la càrrega de comput que no es podria assumir en codi i tan sols ens preocuparíem de la connexió entre SAN i *Wolfram Alpha*.



Imatge 2.10. Resultat d'una consulta a *Wolfram Alpha*

Lamentablement, per accedir a *Wolfram Alpha* a partir d'un programa extern fa falta un registre d'usuari i posterior activació de l'aplicació a través d'un codi. L'aplicació també està restringida a una quantitat de consultes mensuals, pel que ens donava diversos problemes d'accessibilitat de cara a l'usuari final.

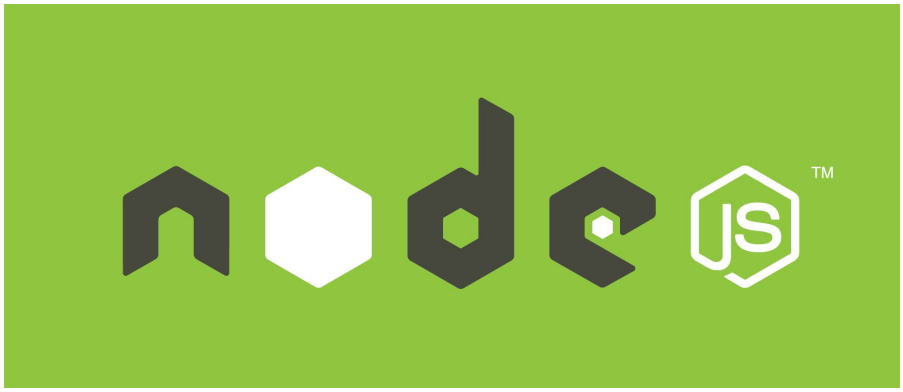
Comunicació amb *Arduino*

Un altre tema a tenir en compte era la connexió amb *Arduino*. Una primera idea va ser la d'utilitzar una placa *Ethernet Shield* que ens permetés la connexió d'*Arduino* mitjançant una connexió Ethernet i l'ús d'una pantalla *LCD* i diversos botons que ens servís d'interfície de control. Tot i això, es va considerar més senzill, pràctic i econòmicament viable de cara l'usuari, el maneig de l'*Arduino* a través d'una interfície al PC.

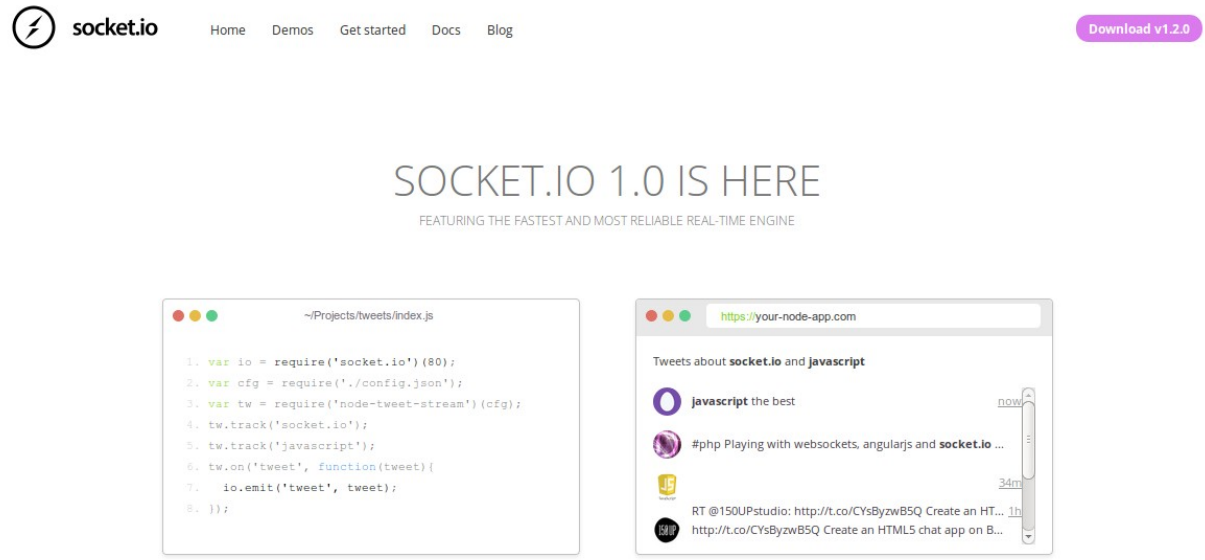
Imatge 2.11. Placa *Ethernet Shield*

Per aquest control, podíem usar la interfície de control del port sèrie que ens ofereix el software d'*Arduino*, però igualment hauríem d'usar la placa Ethernet per comunicar-nos amb la resta del sistema. Per les mateixes raons anteriors, es decideix finalment, crear una interfície pròpia que es comuniqui directament a la resta del sistema.

En primer lloc, es va intentar programar una interfície web basada en **NodeJS**. Es tracta d'un *framework* de *Javascript* que ens permet usar aquest codi per programar la part del servidor. Per la comunicació entre la interfície i el port sèrie d'*Arduino*, es va utilitzar la llibreria de *NodeJS* **SocketIO**, especialitzada en comunicacions entre dispositius i pàgines web. Lamentablement, els resultats no van ser els desitjats: *SocketIO* no funcionava del tot com esperàvem, ens trobàvem amb diversos errors de comunicació amb *Arduino* i la documentació que podíem trobar no ens mostrava cap solució.

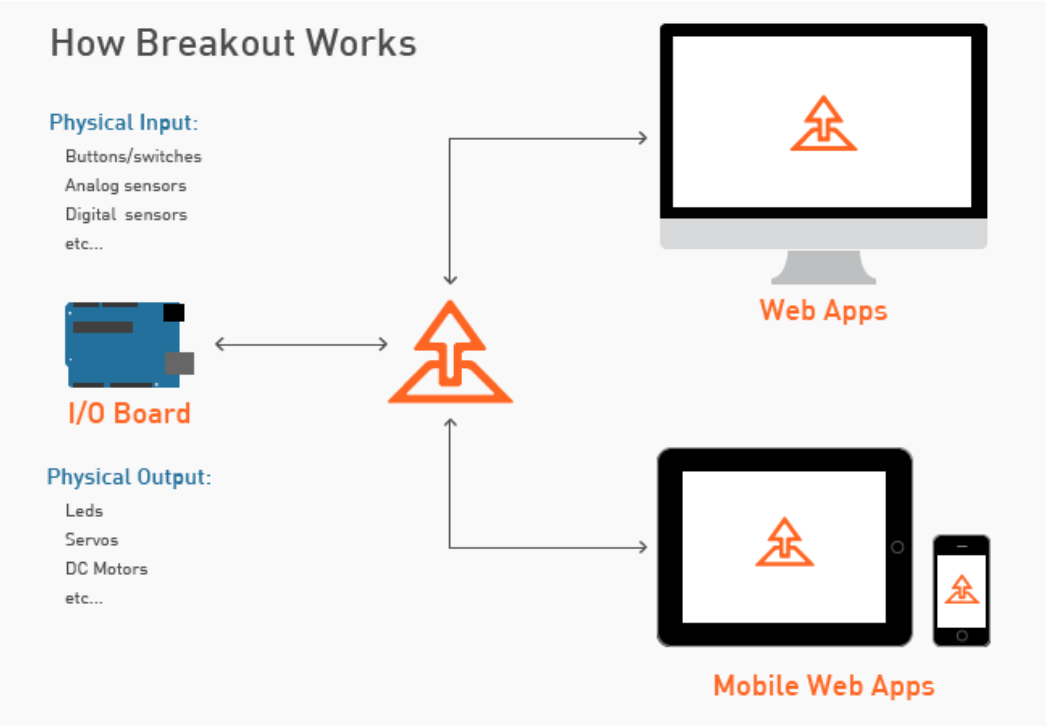


Imatge 2.12. Logo de *NodeJS*



Imatge 2.13. Pàgina principal de *SocketIO*

SocketIO va ser descartat i es van proposar alternatives com **Noduino** o **BreakoutJS**. Aquests dos *frameworks* també estaven basats en *NodeJS* i pretenen ser un pont entre els dispositius i el navegador. Cap d'aquestes dues opcions van donar resultats millors, donat a la poca documentació i errors inesperats. A falta de noves opcions, es descarta realitzar la connexió a través d'una interfície web.



Imatge 2.14. Diagrama de funcionament de *BreakoutJS*

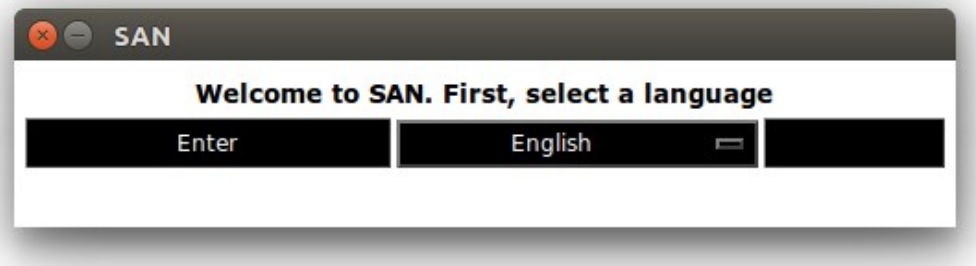
Finalment es va decidir tirar de la llibreria **PySerial** de *Python* per la comunicació amb *Arduino* i implementar una interfície basada en la llibreria de *Python* **Tkinter**. Els resultats obtinguts han sigut molt més prometedors. Tant *Tkinter* com *PySerial* han sigut dues llibreries molt senzilles d'utilitzar i amb una documentació suficient per a resoldre els problemes sorgits.

L'únic inconvenient de *Tkinter* és que es tracta d'una llibreria una mica antiquada, pel que diverses funcionalitats que estan del tot solucionades en altres entorns gràfics, son una complicats d'implementar o directament no funcionen.

D'aquesta manera, s'ha creat un petit programa local. S'han intentar desenvolupar dues versions del mateix programa, una amb *Tkinter* i l'altra per línia de comandes. Degut als requisits de presentació d'informació, s'ha acabat abandonant la versió per línia de comandes.

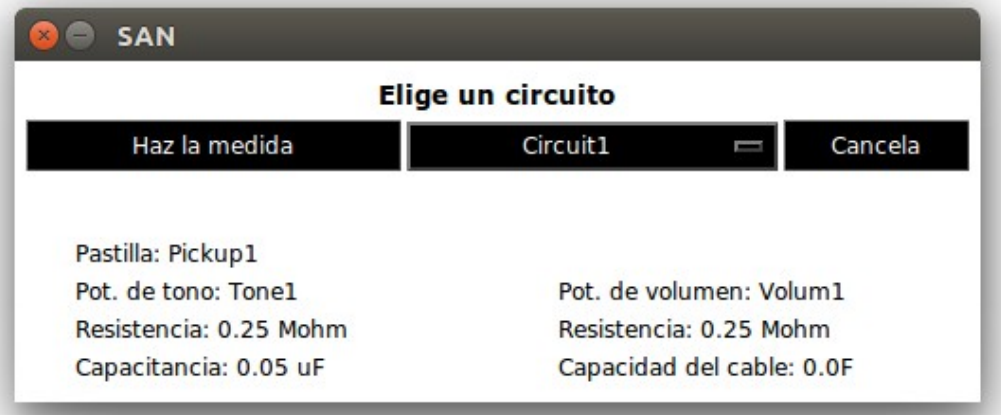
Les dades que tractarem en aquest programa seran els circuits. Donat a que necessitem els valors dels potenciòmetres per el càlcul dels valors de la pastilla, tindrem a la nostra disposició aquells circuits amb una pastilla **buida**, és a dir, sense valors als seus paràmetres.

Primer de tot el programa ens pregunta sobre l'idioma que volem fer servir (català, castellà o anglès).



Imatge 2.15. Aplicació local SAN, pantalla inicial

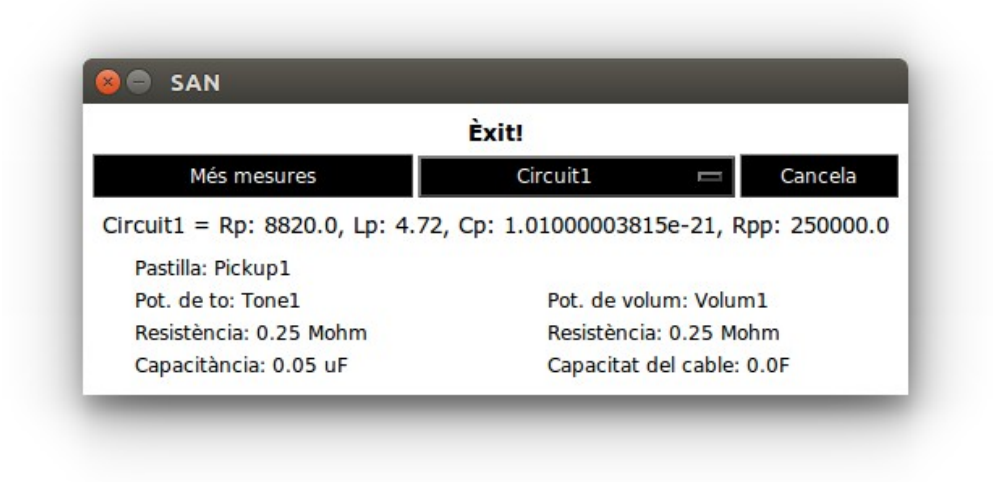
Acte seguit, un desplegable ens mostra els circuits disponibles a ser mesurats. Al triar un circuit, se'ns mostrarà la informació dels components d'aquest circuit, incloent el nom de la pastilla que conté. Aquesta informació era molt extensa per ser presentada per línia de comandes; aquesta és una de les raons per les que s'ha abandonat aquesta versió.



Imatge 2.16. Aplicació local SAN, pantalla de selecció de circuit.

Un cop estem segurs que hem triat el circuit que volem i tenim l'*Arduino* preparat per la mesura, indiquem al programa que volem començar la mesura. Es mesuraran els valors estàtics de les pastilles. Pel càlcul del volum, el programa ens demanarà que connectem el circuit de mesura del volum i toquem la tercera corda (G) de la guitarra. Un cop fet això, s'enviaran els resultats a SREC. Un cop s'hagi mesurat el circuit, aquest quedarà invalidat per tornar a ser mesurat. Com que el càlcul ha estat basat sobre els valors de la pastilla, aquests son independents del circuit on es trobi. Per tant, tots els circuits no mesurats que també tinguin aquesta pastilla, seran invalidats. Així, si tenim una pastilla **P1**, dos circuits **C1** i **C2** i els dos tenen la pastilla **P1**, el càlcul sobre el circuit **C1** farà que s'invalidi **C1** i **P1**, però també **C2**.

Per últim, el sistema ens preguntarà si volem fer més mesures o tancar el programa. També podem cancel·lar una opció i tornar al pas inicial sempre que volem.



Imatge 2.17. Aplicació local SAN, pantalla d'èxit de mesura

Aquest programa ha estat desenvolupat sobre **Ubuntu 12.04** i la versió 2.7 de *Python*. Per tal que l'usuari el pugui fer servir, se li proporcionarà el programari requerit. Farà falta la instal·lació de *Python* 2.7, la llibreria de comunicació amb el port sèrie *PySerial* i *Tkinter*. Es recomana fer servir l'instal·lador de paquets *Python PiP*. Per pujar el codi d'*Arduino*, també ens faltarà descarregar-nos el seu software. El programari es pot trobar a aquestes direccions:

```
www.python.org/download/releases/2.7/
pypi.python.org/pypi/pip
pyserial.sourceforge.net/
tkinter.unpythonic.net/wiki/How_to_install_Tkinter
arduino.cc/en/pmwiki.php?n=main/software
```

2.5. Sistema de disseny (SED)

2.5.1. Introducció

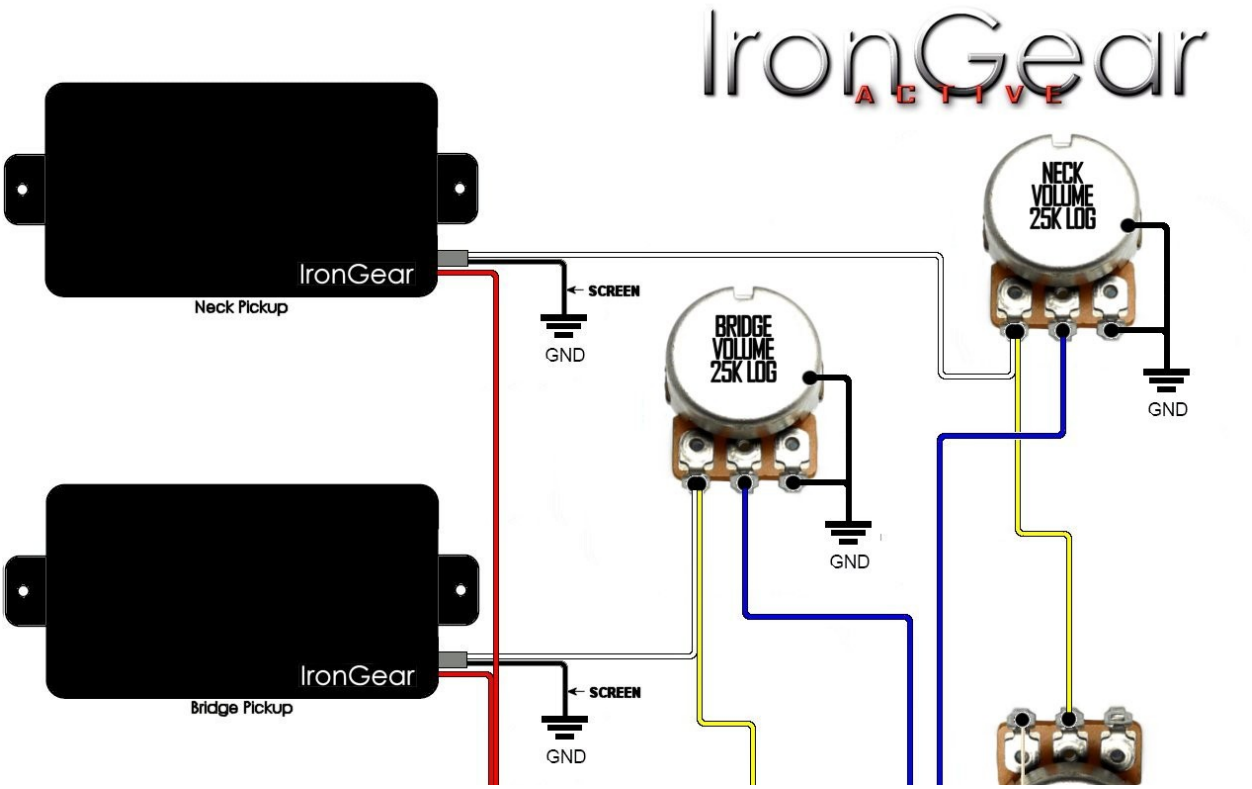
SED és el subsistema de disseny de circuits. La primera intenció de SED era que l'usuari tingués la capacitat per poder representar gràficament el circuit que l'usuari volgués, donant un ampli ventall de components i configuracions. Finalment, aquesta idea s'ha anat simplificant.

2.5.2. Prototips

Com ja hem dit, la primera idea era donar una gran llibertat a l'usuari perquè dissenyés el seu circuit. L'imatge mental que es tenia pensada, era el disseny a partir d'una interfície gràfica, on poder agregar-hi nous components que tinguéssim a una llibreria i connectar-los entre si.

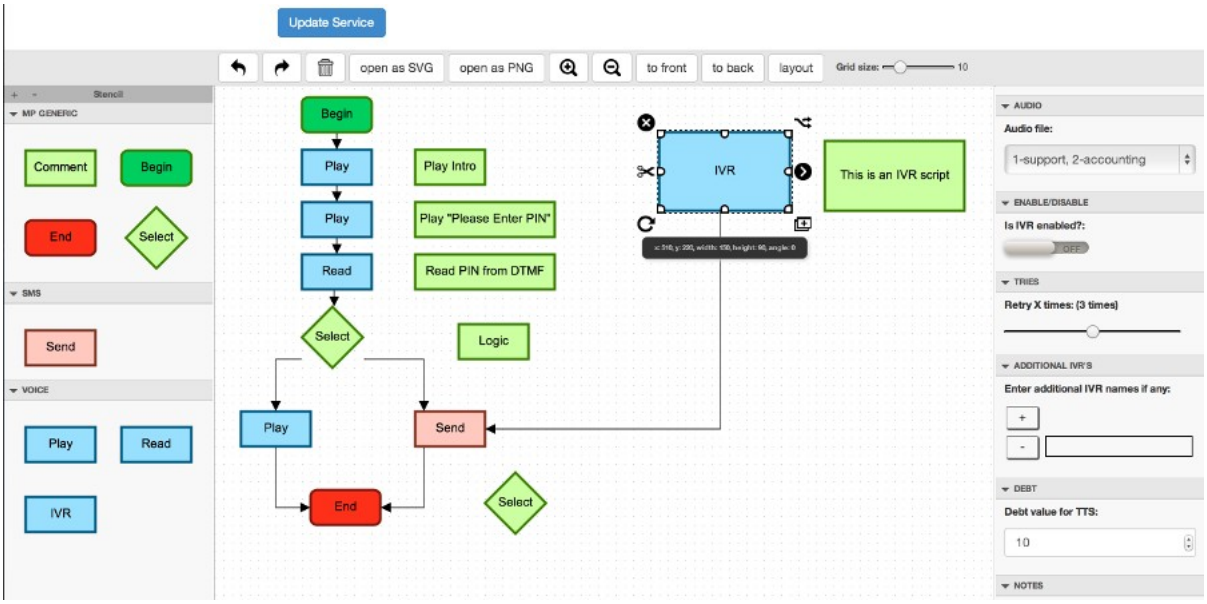
A la pàgina web de la marca de pastilles *Iron Gear*, a la secció *Wiring Diagrams*, podem trobar una sèrie de diagrames de circuits. Els diagrames son configuracions típiques que no només ens donen una idea gràfica de representar el circuit que volem, sinó el llistat de components típics que podem trobar.

Així doncs, tenim una bona base per començar a preparar SED.



Imatge 2.18. Exemple de diagrama *Iron Gear*

Principalment, es va partir de la implementació directa de diagrames amb una aplicació web basada en la llibreria de *Javascript* **JointJS**. És una llibreria orientada al dibuix i manipulació de diagrames. A partir de senzilles funcions, som capaços de crear polígons o visualitzar imatges, moure'ls per la nostra àrea de dibuix o enllaçar-los a partir de cables, que també poden ser manipulats. Alguns exemples de *JointJS* son un joc d'escacs o aplicacions de diagrames *UML*, entre d'altres.



Imatge 2.19. Editor de diagrames de flux fet amb *JointJS*

Així doncs, definirem uns casos d'ús molt senzills:

Crear un nou projecte: Volem crear un nou projecte, representat per una àrea de dibuix en blanc, que serà on disposarem el nostre circuit. En aquest projecte li assignem un nom demanat pel sistema.

Crear un nou element: Dins el nou projecte, hi podem agregar nous elements disponibles a una llista. Aquesta llista conté pastilles senzilles, pastilles *humbucker*, condensadors, potenciòmetres de to i volum, palanques de *switch*, preses de terra i altres elements. A cada element li assignem un nom i aquest ens sortirà representat a través d'una imatge sobre la nostre àrea de dibuix. Al costat de l'àrea disposem d'una llista dels elements que tenim, tot incloent el seu nom.

Enllaçar un nou element: Existeix l'opció d'enllaçar dos elements. Per això, elegim dos elements d'un parell de menús desplegables i indiquem que volem enllaçar-los. A cada enllaç o cable s'inclourà un nom i apareixerà de la mateixa manera a l'àrea de dibuix i a la llista d'elements.

Eliminar un element o enllaç: A la llista d'elements tenim l'opció d'eliminar l'element desitjat. Si es tracta d'un enllaç, aquest simplement s'elimina. Per tal de mantenir l'estructura, si es tracta d'un altre tipus d'element, s'eliminarà l'element i els enllaços que tingui assignats.

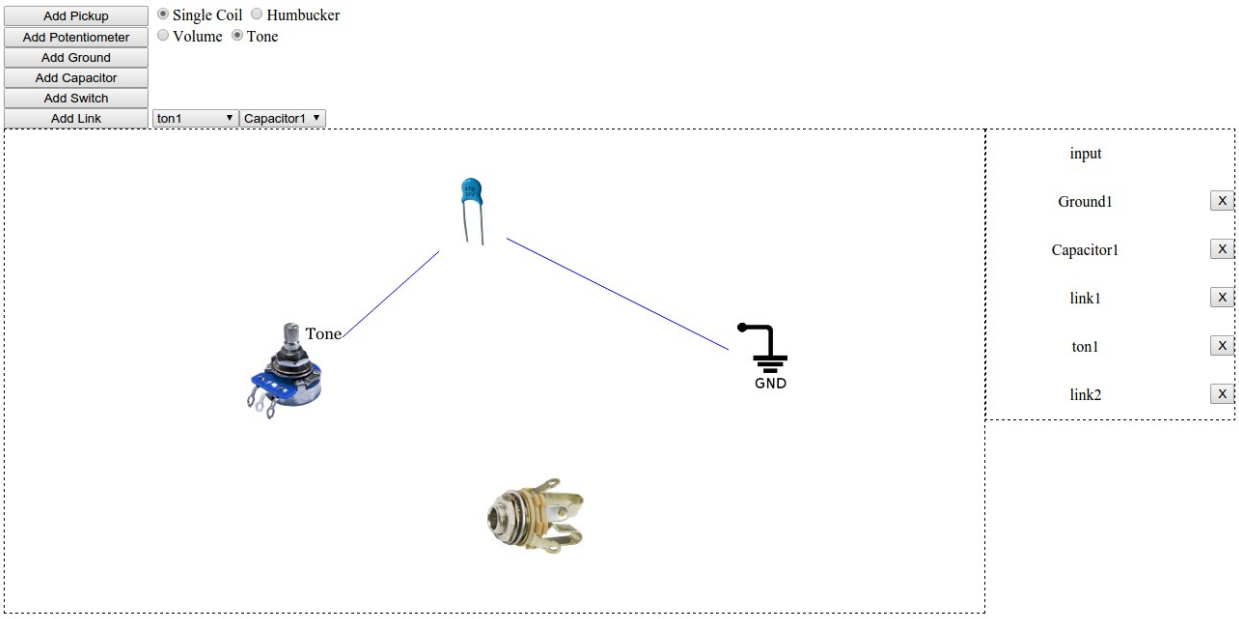
Gràcies a *JointJS*, sense necessitat d'especificar-ho, podem moure els elements per la zona de dibuix i distorsionar la forma dels enllaços.

No obstant, aquesta iniciativa va ser apartada del projecte donat a la falta de precisió de l'aplicació. El programa donava la capacitat per ajuntar diversos elements a través de cablejat però li faltaven restriccions i, sobretot, un mecanisme per identificar els diferents enclavatges de cada element. Si ens

fixem en un dels diagrames d'*Iron Gear*, veurem com el cablejat de cada element és diferent, així com el nombre d'enllaços que un element pot tenir. Això és una cosa que no podem controlar amb *JointJS*, almenys directament a través de la llibreria. Per exemple, un potenciòmetre consta de tres ranures. Un podia connectar una pastilla amb el potenciòmetre però no hi havia manera gràfica de triar quina de les tres ranures s'escollia.

Una proposta hagués sigut especificar classes per cada tipus d'element i restringir el nombre d'enllaços que es podien tenir. S'enumeraria cada ranura disponible i, al moment d'enllaçar dos elements, el sistema demanaria en quina ranura es volia connectar. No obstant, gràficament no haguéssim vist cap canvi i, a part, la construcció lliure d'un circuit implicaria que es podrien generar una sèrie de diagrames incorrectes. Això seria, per exemple, un circuit sense pastilla, diversos potenciòmetres de volum sobre la mateixa pastilla o un circuit sense sortida.

El programa no va passar d'un primer prototip.



Imatge 2.20. Prototip de SED amb *JointJS*

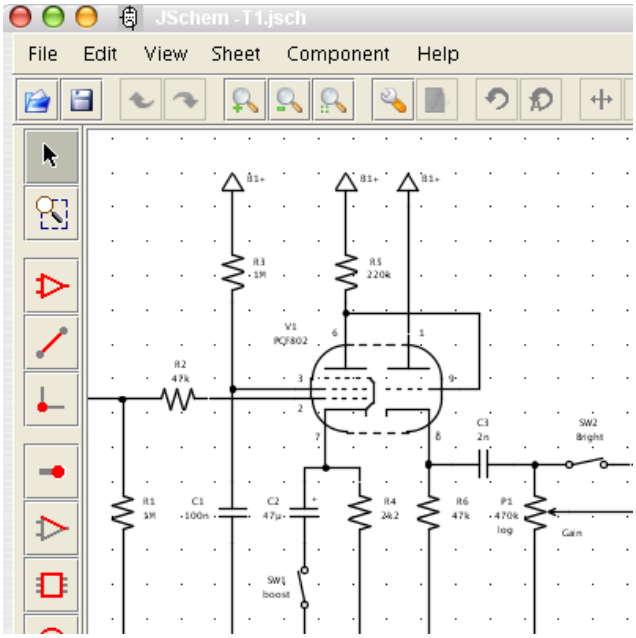
Es va considerar que la implementació d'un sistema gràfic que cobrís aquesta precisió implicava una càrrega de treball prou alta com per dedicar-hi tot un projecte, pel que es va decidir tirar cap a la integració de software de disseny ja implementat.

Per tant, ens faria falta un programa que ja cobrís l'edició gràfica amb prou precisió, que tingués la capacitat d'incloure elements propis i que poguéssim integrar a un sistema a part.

Alguns programes a considerar van ser:

Jschem. Aquest programa està orientat al disseny de circuits electrònics sobre una quadricula. No té molta qualitat visual però té una gran capacitat per crear nous elements i guardar-los. *Jschem* també dona la capacitat d'exportar els circuits en format XML, pel que resultava interessant poder treballar amb un format tant manejable. *Jschem* pot ser modificat a partir del seu codi font, disponible a la seva pàgina web. Aquest és un projecte d'*Eclipse*.

Lamentablement, la creació de nous elements resultava massa precisa i, el mínim desviament donava lloc a elements molt diferents. *Jschem* basa la creació de nous elements en l'agrupació de tipus bàsics de la seva llibreria. Aquesta agrupació es fa a nivell gràfic sobre una quadrícula.

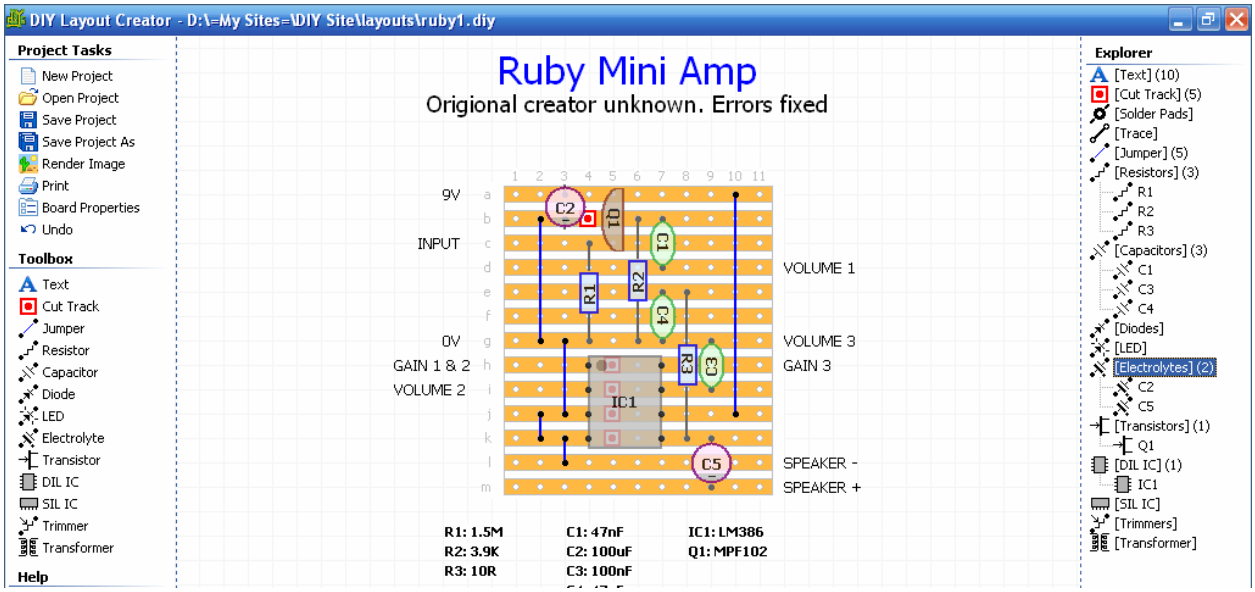


Imatge 2.21. *Jschem*

Imaginem que volguéssim crear una pastilla. Seguint els diagrames d'*Iron Gear*, aquesta tindria un format quadrat i tindria 3 ranures (representades per 3 punts o 3 cercles). Ens interessa posar els punts propers al quadrat, perquè s'entengui que formen part de la mateixa figura. Això implicava posar-los tocant-se amb un dels costats del quadrat. Donat a la poca precisió que tenim amb el cursor del PC, això dona lloc a punts massa allunyats, massa cèntrics o massa enfonsats al quadrat. En tots tres casos, el resultat de l'XML era diferent.

DIY Layout Creator. Aquest programa està altament orientat al disseny de circuits de tot tipus a un nivell més alt que *Jschem*. Té una qualitat visual més gran i una llibreria especialitzada en elements de guitarra. Així, podem trobar la majoria d'elements típics de guitarra ja configurats. Amb *DIYLC* podem enllaçar elements però tenim el mateix problema que amb la nostra aplicació de *JointJS*. Ens fa falta una especificació de ranures i un comprovador que ens indiqui la correctesa del nostre element.
Les dades tampoc poden ser exportades en formats manipulables, només obteníem l'opció d'exportar el nostre projecte a un PDF.

Cap dels dos programes semblaven poder cobrir les nostres necessitats degut a la quantitat d'inconvenients i poca integració que oferien. D'altra banda, al sistema SAN teníem que la mesura del circuit només podia donar-se a partir d'un circuit senzill de pastilla, volum i to i que afegir nous elements complicaria desmesuradament aquest càlcul, pel que SED hauria de reduir les seves funcionalitats.



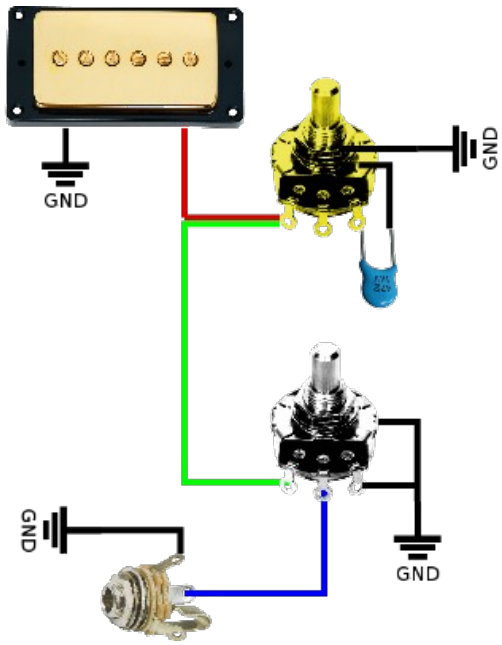
Imatge 2.22. *DIY Layout Creator*

2.5.3 Canvas i HTML5

Finalment es decideix realitzar una visualització gràfica molt simplificada. L'usuari podrà triar entre diferents imatges per cada un dels elements creats que composen el circuit. Al dissenyar un circuit, es farà una petita visualització de les imatges dels components integrats sobre un cablejat estandarditzat. Per cada una de les configuracions de circuits existents, es dibuixarà un circuit o altre. Les imatges usades també han estat estandarditzades, pel que la posició del cablejat sobre les diferents ranures dels components és prou precisa.

Dins la pàgina de creació d'un circuit, l'usuari triarà els diferents components del circuit, tot depenent de la configuració triada. Aquests s'aniran dibuixant, junt amb el seu cablejat quan siguin seleccionats. Algun cablejat forma part de diversos components, pel que s'ha intentat repartir el nombre de cables entre els diferents components.

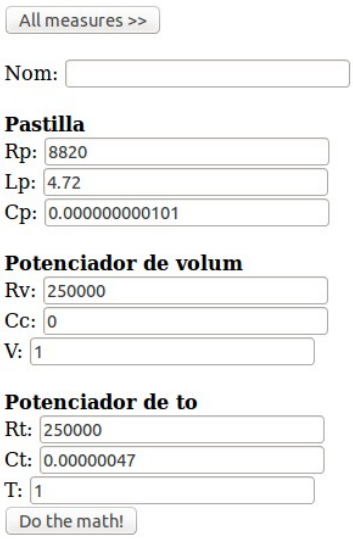
Per fer això, hem utilitzat *jQuery* i l'element de dibuix *<canvas>* d'*HTML5*. Es tracta d'un llenç sobre el que podem dibuixar-hi línies, objectes geomètrics o incrustar-hi imatges. En el nostre cas hem utilitzar el traçat de línies per representar el cablejat del circuit i la incrustació d'imatges pels components. La dificultat més gran que té *<canvas>* és que hem de re-dibuixar tot el llenç cada cop que incloem un component nou. En la configuració *serial_1_2* no suposa un greu problema perquè només comptem amb 3 elements, però pot ser una càrrega per a altres configuracions més elaborades.



Imatge 2.23. Representació d'un circuit *serial_1_2* sobre *canvas*

2.5.4. Evolució de SED

Pel desenvolupament de SED es va seguir un sistema iteratiu, de manera que s'anés progressant en les seves funcionalitats.



Imatge 2.24. SED, versió 1

En la primera versió, SED era tant sols una interfície web on entrar-hi els valors d'un circuit. Aquestes dades s'usaven per fer les mesures dinàmiques (SAN) i s'emmagatzemava tot dins una sola classe *Measure* (mesura).

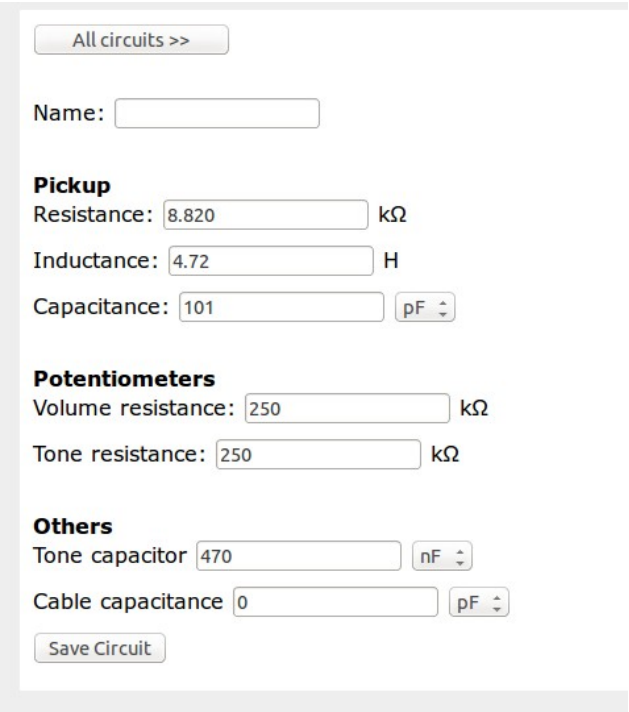
En la segona versió, SED ja inclou un mòdul de circuit amb les dades de les pastilles, potenciòmetres, condensador de to i cablejat i està separat de les mesures. Llavors el que tenim és que un circuit pot tenir diverses mesures. Hem considerat que el circuit és estàtic i el que canvia d'una mesura a l'altra és la posició dels potenciòmetres. Així podem tenir diverses mesures sota els mateixos paràmetres i, canviant tan sols un parell de paràmetres, obtenim noves mesures que poden ser fàcilment contrastables. Per tant tindríem la classe *Measure* i *Circuit*.

En aquest cas, la classe *Measure* deixa de tenir nom i s'identificarà pel seu id (creat automàticament) i el nom del circuit sobre el qual s'ha realitzat la mesura.

La tercera versió ja inclou 2 mòduls més, que corresponen a la classe *Pickup* (Pastilla) i *Potentiometer* (Potenciòmetre). Per cada mòdul tenim una pàgina diferent i cada element serà identificat per un nom assignat per l'usuari.

Per la classe *Pickup*, hi entraran els valors estàtics d'una pastilla. Quan creem una pastilla a SED, aquesta no tindrà valors pels seus paràmetres, doncs aquests seran obtinguts mitjançant SAN. L'únic que podem escollir a SED és el nom de la pastilla i una imatge que la representi visualment.

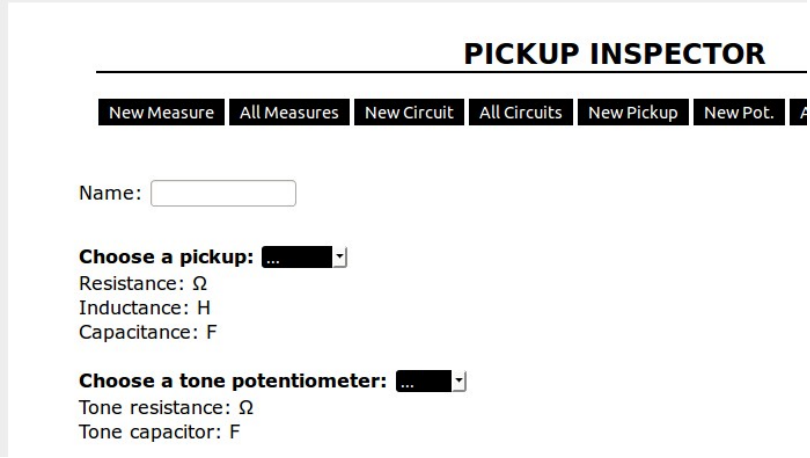
La classe *Potentiometer* inclou la creació dels dos tipus de potenciòmetre: de to i de volum. En els dos casos s'inclourà el nom del potenciòmetre i una referència gràfica. En cas de seleccionar un potenciòmetre de to, haurem d'incloure el valor de la capacitat condensador. Hem decidit no separar un element de l'altre ja que el condensador de to aporta poca informació addicional al circuit i haver de crear un element a part tan sols per el valor de la seva capacitat, no tenia molt sentit.



Imatge 2.25. SED, versió 2. Creació d'un circuit

En canvi, excloem el potenciòmetre de volum d'aquesta decisió (el potenciòmetre de volum tant sols té el valor de la seva resistència) ja que forma part de la visió global del circuit:

L'usuari pot identificar els elements d'un circuit fent una ullada a la seva guitarra. Veurà que consta d'una pastilla i de dos potenciòmetres, però no veurà el condensador, doncs aquest està lligat al potenciòmetre de to, però si que veurà el potenciòmetre de volum. D'altra banda, la senzillesa, tant pràctica com conceptual que ens ofereix poder crear potenciòmetres de to i volum sense canviar de pàgina no seria equiparable a afegir el valor de la resistència del potenciòmetre de volum en una altra pàgina o classe (per exemple, si aquest valor estigués lligat a la classe *Pickup* o s'especifiqués directament al circuit).

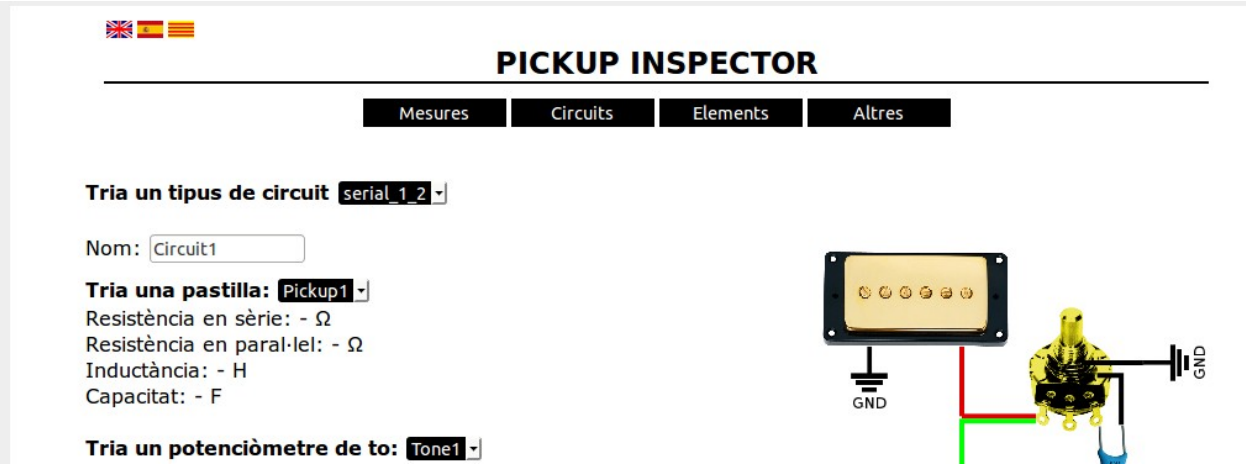


Imatge 2.26. SED, versió 3. Creació d'un circuit

Degut a aquestes dues noves classes, el paradigma de disseny d'un circuit es modifica. Ara tindrem, a la pàgina de creació d'un circuit, tres llistes d'elements (pastilla, to i volum). El que si que incloem al circuit és el valor de la capacitat del cable perquè és tracta d'un valor que és opcional. Aquest valor no afecta en excés a la mesura i, per tant, no té una gran rellevància com per està disposat com un element a part. A més, conceptualment, no es pot incloure a cap altre element com hem fet amb el condensador de to. Per defecte, aquest valor és 0.

Aquesta tercera versió ens permet disposar d'una gran varietat d'elements en poc temps i crear els circuits de manera molt més ràpida sense entrar cada un dels valors de cada element, com ens podia passar en la primera o segona versió. L'únic inconvenient és tenir una llibreria escassa. Si tenim el sistema buit o no hi consta cap dels elements que volem, haurem de crear-los un per un, pel que pot esdevenir una feina feixuga.

A la quarta versió mantenim l'estructura de l'anterior però podem visualitzar la representació del nostre circuit. El sistema ens permetrà triar, per cada un dels components del circuit, una imatge que el representi gràficament.



Imatge 2.27. SED Versió 4. Creació d'un circuit

Per últim, a cada una de les versions s'hi ha inclòs un comprovador, que valida que els valors entrats per l'usuari son correctes. En cas de no ser-ho, SED ens avisarà. Aquestes validacions son:

- El nom de la mesura és únic al sistema (només versió 1).
- El nom de la mesura no és buit i és alfanumèric (només versió 1).
- El nom de la mesura no té més de 20 caràcters (només versió 1). *

- El nom del circuit és únic al sistema.
- El nom del circuit no és buit i és alfanumèric.
- El nom del circuit no té més de 20 caràcters. *

- El nom de la pastilla o potenciòmetre és únic al sistema (només versió 3).
- El nom de la pastilla o potenciòmetre no és buit i és alfanumèric (només versió 3).
- El nom de la pastilla o potenciòmetre no té més de 20 caràcters (només versió 3).*

- Els valors de la pastilla son numèrics i no son negatius.
- Els valors de la pastilla no son buits (versió 1 i 2).
- Si no s'indica el contrari, els valors de la pastilla no son buits (només versió 3).

- Els valors dels potenciòmetres no son buits, son numèrics i no son negatius.
- Els valors de les posicions dels potenciòmetres estan entre 0 i 1**.
- El valor del potenciòmetre de volum no pot ser 0 (versió 2 i 3).

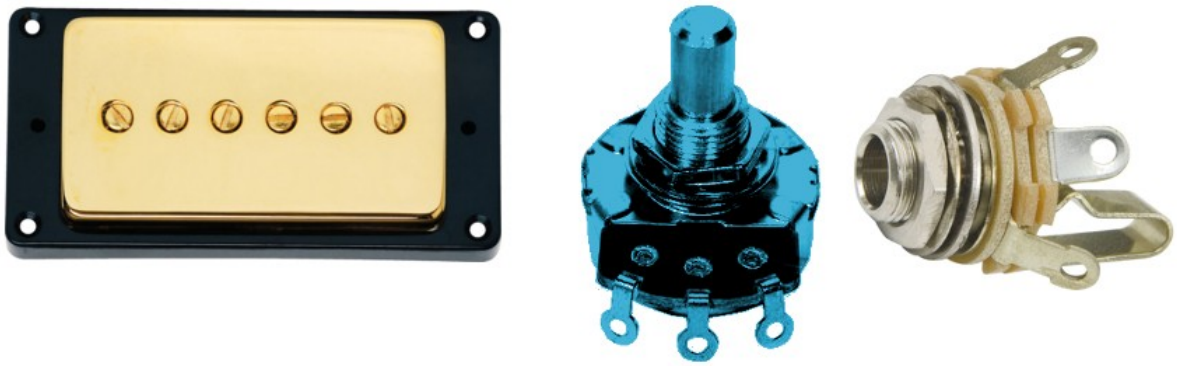
* Aquesta restricció s'ha afegit per mantenir certa llegibilitat i estructura dins la visualització a SREP. Donat a que el sistema no s'ha pensat per ser escalable, creiem suficient la combinació de 20 caràcters alfanumèrics per expressar els diferents elements que puguin haver-hi.

**Aquesta validació no té sentit en l'última versió del programa, doncs els valors de les posicions son entrats per l'usuari a partir d'un objecte *slider* que ja restringeix els valors possibles.

Per últim, a partir de la segona versió, s'ha inclòs una pestanya per triar l'escala de valors que volem fer servir. Pensada en un principi per cada un dels valors a entrar, es va veure que només tenia sentit triar valors quan es tractava de les capacitats.

Les resistències sempre ronden els kOhms, pel que és natural entrar els valors en aquesta mesura i no en Ohms. La inductància sempre entrarà en *Henries*, així que no hi ha res a triar. Les capacitats solen estar expressades en pF (10^{-12}) o en nF (10^{-9}) o en µF (10^{-6}) pel que s'ha donat l'opció de triar entre aquests tres exponents.

Aquesta funcionalitat dona més facilitat d'ús a l'usuari que entra els paràmetres, doncs evita fer càlculs externs al programa. Els pocs casos que un valor pot necessitar d'una escala major o menor son casos residuals que no ocasionen un problema a la majoria d'usuaris.



Imatge 2.28. Algunes imatges per representar una pastilla, un potenciòmetre i la sortida del circuit, respectivament, a la versió 4 de SED.

2.5.5. Software

El codi usat per aquestes pantalles ha estat codi d'estructura web (*HTML5*, *CSS*, *Javascript*) per a la visualització de les pàgines i codi *Javascript* amb llibreries *jQuery* amb mètodes *AJAX* per fer les comprovacions, canvis d'escala i comunicació amb SREC.

2.6. Sistema de recol·lecció (SREC)

2.6.1. Introducció

SREC és el subsistema més necessari però menys visible. En realitat es tracta d'un entramat de mètodes i funcions que integren la resta de subsistemes i fan la comunicació entre ells. Per això, SREC inclou la persistència de les dades, la comunicació amb SAN i SREP i altres funcionalitats.

2.6.2. Integració

Es pretén la integració dels subsistemes dins la mateixa plataforma. D'aquesta manera la comunicació és directa i l'accés de l'usuari és la mateixa per tots els subsistemes. És per això que les dues opcions explorades son:

- Una aplicació web, amb connexió a un servidor, accessible des d'Internet.
- Una programa local.

Un programa local ens hagués permès l'ús de llenguatges més potents, com C, però haguéssim hagut de treballar amb interfícies gràfiques de les que no en tenim massa coneixement.

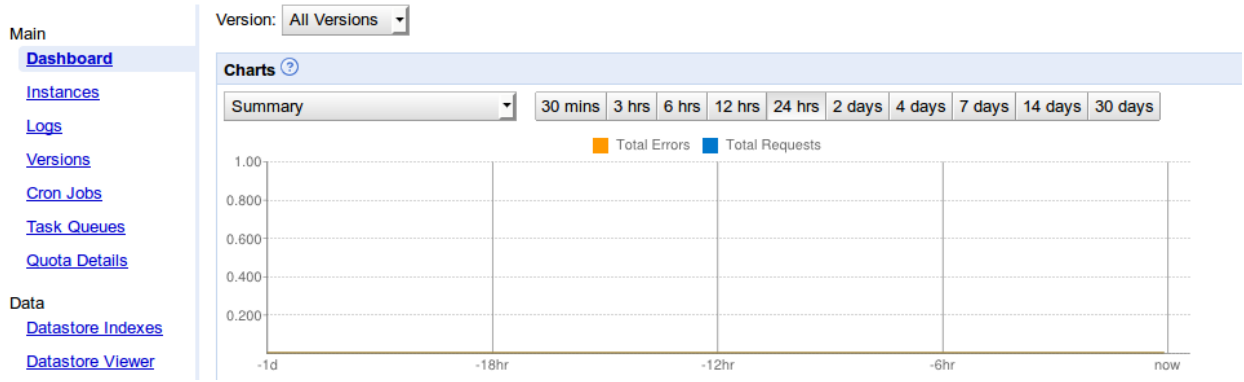
Per accessibilitat amb l'usuari i experiència en interfícies web, es va decidir treballar amb la primera opció. Això implica que l'usuari no necessita una gran configuració per usar el sistema, però a la vegada implica que necessita necessàriament d'una connexió a Internet. Això no és un problema actualment, doncs l'accés a Internet és pràcticament global.

També trobem el problema que l'accés és global, pel que qualsevol usuari que arribi a aquella direcció podrà actuar sobre el nostre sistema. Això és un problema que es pot resoldre fàcilment amb un sistema d'usuaris però que per càrrega de treball no s'ha realitzat. Queda pendent a les millores.

La plataforma triada per fer de servidor ha estat *Google App Engine*. Aquest servei de Google està especialment orientat a aplicacions de caràcter genèric i conté unes prestacions prou rellevants:

- *Google App Engine* és gratuït.
- És un servei que permet la implementació del nostre servidor en *Python*, *Java*, *PHP* o *Go*.
- L'allotjament al servidor inclou una direcció de domini a l'estil **nomdelaplicacio.appspot.com** pel que facilita l'accés remot.
- Té suport per 500 *megabytes* d'emmagatzemament i suficient CPU i ample de banda per milions de visites mensuals.
- *Google App Engine* usa per defecte el *framework* **webapp2** per *Python*. *Webapp2* és un entorn de treball molt senzill d'utilitzar.
- Al ser una eina de Google, conté una gran documentació per resoldre dubte si errors que poguessin aparèixer, així com un tutorial per a principiants.
- Conté eines d'anàlisi i gestió de la nostra aplicació com estadístiques, consulta a la base de dades o control de versions.

Així doncs, *Google App Engine* ens dona prou raons com per triar-la com a plataforma per SREC.



Imatge 2.29. Google App Engine

2.6.3 Comunicació

SREC fa la feina de rebre les dades que ens arriben des de SAN i SED i les peticions d'SREP. Així doncs, SED serà proveït pel servidor quan volem accedir a ell i guardarem els elements instanciats quan l'usuari ho indiqui.

La nostra intenció era que SAN estigués igualment inclòs a *Google App Engine* però resulta que certa llibreria necessària per usar la llibreria *PySerial* de *Python* (*fcntl*) és de les poques llibreries que el servei de Google no accepta. Finalment, la part de SAN que si que està inclosa al servidor és la mesura dels valors dinàmics, doncs es tracta d'una funció implementada amb *Javascript*.

Així doncs, la comunicació que hem de fer amb el programa local SAN (programat igualment amb *Python*, com ja s'ha dit) és a partir d'una connexió amb el protocol HTTP. El programa local de SAN utilitza la llibreria **url2** de *Python* per demanar a SREP les pastilles a mesurar i per comprovar la unicitat del nom d'una nova pastilla, així com per enviar els valors estàtics mesurats. Per proveir les dades a l'SREP, funcionarem de la mateixa manera.

Per la comunicació entre el servidor i el client i el programa i la pàgina web s'ha estandarditzat l'enviament d'arxius JSON, que contenen, per cada paquet, el contingut de l'objecte tractat o una llista de valors. Per cada instància d'una classe existeix un mètode *serialize* que ens permet passar tot un objecte en format JSON. La majoria de casos no requerim de tots els atributs però és un bon hàbit oferir-los per possibles millores sense necessitat de canviar excessivament el codi.

2.6.4 Peticions i funcionalitats

Per simplificar les peticions al servidor, les hem estructurat de la següent manera:

Totes les peticions provenen d'una dada concreta d'una classe. És per això que, si demanem una dada sobre una pastilla o se'n vol modificar el seu contingut, l'*url* de la petició començarà per */pickup*. Tenim 4 classes on es segueix aquesta estructura: *pickup* (per pastilles), *pot* (per potenciòmetres), *measure* (per mesures) i *circuit* (per circuits).

Després d'especificar la classe, apareix el nom del que serà la funcionalitat a realitzar. Depenent de la classe hi haurà unes funcionalitats o altres.

En general tenim les següents funcionalitats:

- Accés a SED (*/new*). Se'ns proporciona d'interfície gràfica per introduir els valors.
- Creació (*/save*). Es crida des de SED, és la funció que comprova la correctesa de l'element i la que envia les dades al SREC perquè les guardi a la BD.
- Càrrega (*/load*). Aquesta funcionalitat permet carregar les dades demanades. S'usa normalment per a llistes desplegables. Per exemple, quan volem crear un circuit nou, ens farà falta cridar a */pickup/load* i */pot/load* per emplenar les llistes desplegables per més tard, poder triar un d'aquests valors.
- Llistat (*/list*), també és una funcionalitat de càrrega, però està orientada a servir SREP. Mentre */load* normalment només serveix llistats d'identificadors d'elements, */list* entrega més dades. Aquesta funcionalitat no està inclosa a la versió 3 per pastilles i potenciòmetres, doncs aquests elements es disposen a la mateixa pàgina. En aquest cas, s'ha decidit usar una classe auxiliar */element* amb l'única funcionalitat */list*, que ens retornaria un llistat tant de potenciòmetres com de pastilles.
- En el cas de mesures i circuits, la funció */list* pot contenir un paràmetre més que representa l'identificador de l'element tractat. En aquest cas, el que ens mostrarà la funcionalitat serà la visió en detall de l'element. Això està explicat a l'apartat d'SREP.
- Eliminació (*/delete*). Es refereix a l'eliminació d'un element concret. Aquesta opció està explicada a l'apartat d'SREP.

Per carregar els valors dels circuits no mesurats a l'aplicació local del SAN, s'utilitzarà la funcionalitat *circuit/list_measured*.

També tenim la funcionalitat */update* per circuit, que també es fa servir a SAN. Aquesta funcionalitat és la que permet modificar un circuit. Donat a que des del SAN només rebem circuits amb pastilles 'buides' (és a dir, sense valors, pendents a mesurar), les pastilles i circuits que es modificaran son els que encara no tenen valors.

Per últim, tenim peticions per arribar a pàgines estàtiques. La petició per defecte (/) ens porta a l'index de l'aplicació web, des d'on podem accedir a la resta de l'aplicació. A part d'aquesta tenim la funcionalitat */wiki*, que ens porta a la documentació. Aquesta pot tenir una variable que ens indica l'idioma en el qual volem que se'ns presenti la documentació, /en (anglès, *English*) o /cat (català). L'última petició és */source*, que ens serveix una pàgina amb software extra.

2.6.5. Base de dades

NDB

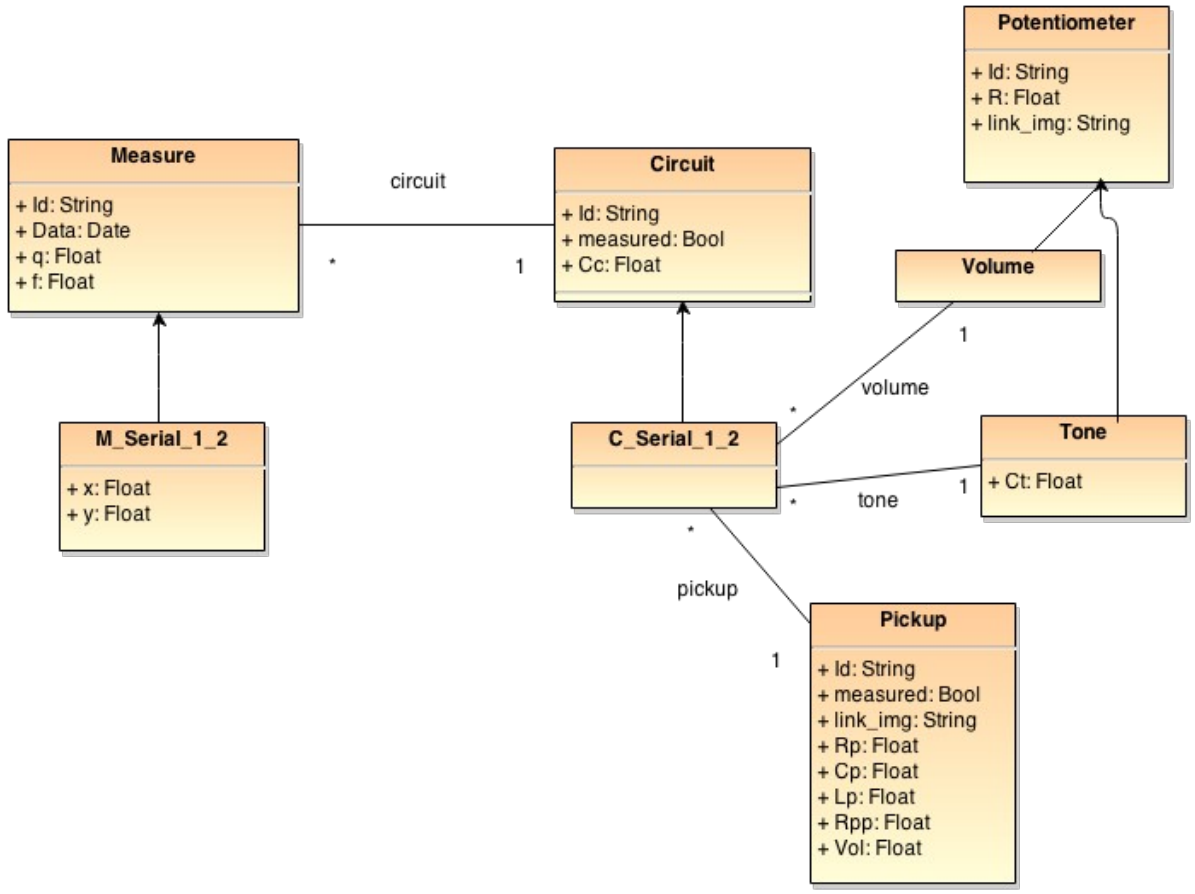
Per implementar una base de dades dins *Google App Engine*, aquest ens ofereix el sistema **NDB**. *NDB* funciona més com una *API* sobre la que realitzar-hi les consultes que com una base de dades a la que estem acostumats (*SQL*).

Primer de tot, el concepte de *NDB* és molt diferent a *SQL*. En aquesta base de dades tenim la informació d'objectes que coneguts com **entitats** (*entities*). Cada entitats conté valors anomenats **propietats** (*properties*) que son tipus definits per l'*API* d'*NDB*. Aquestes propietats poden ser tipus simples (*String*, *Integer*, *Float*,...) així com tipus més complexes com objectes d'usuari, estructures *JSON* o posicions geogràfiques.

Per definir entitats homogènies s'usa el concepte de **Model**, que ve a ser el mateix que una classe a *SQL*. A part d'això, cada entitat està identificada per una **clau** (*key*) que fa la entitat única al sistema. Una clau està formada per un identificador i per un **tipus** (*kind*) que, en el nostre cas, en tenim prou en dir que equival al model de l'entitat. Les claus poden estar jerarquitzades per altres claus, els seus **pares** (*parent*). Aquests pares poden tenir-ne d'altres fins arribar a la **clau arrel** (*root key*). Quan diverses entitats tenen la mateixa arrel, formen un **grup d'entitats** (*entity group*). De cara a la nostra base de dades, el concepte de pare, arrel i grup d'entitats no seran necessaris.

Mapa de base de dades

El mapa de la base de dades ha evolucionat a mesura que s'afegien els mòduls, com hem explicat a SED. Per evitar repetir-nos, tan sols explicarem l'estructura de la base de dades a l'última versió.



Imatge 2.30. Diagrama UML de la base de dades

Els models que hem fet servir son els següents:

Mesura (*Measure*). Representa la classe principal de les mesures que es realitzen sobre un circuit concret. Les entitats de Mesura tenen les següents propietats:

- *Name*. Es tracta d'una propietat *string* que guardarà el nom del circuit sobre el que hem fet la mesura. Aquesta propietat ens servirà de referencia per accedir a les propietats del circuit un cop consultem la mesura.
- *Date*. Es tracta d'una propietat *Date* que guardarà un *timestamp* del moment en què es va fer la mesura. Ens serveix com una dada addicional per identificar una mesura. Aquesta opció està especificada amb l'opció *auto_now_add*, de manera que s'afegeix automàticament al instanciar una mesura.
- *F*. La freqüència de ressonància de la mesura es guarda com un *Float*.
- *Q*. És la qualitat de la mesura, guardada com un *Float*.

Cada mesura vindrà donada per un circuit amb una configuració concreta. Donada la definició que hem fet d'una mesura, aquesta tindrà una sèrie de paràmetres extres que dependran de la configuració del circuit mesurat. Per a cada configuració de circuit, existirà una subclasse de mesura (és a dir, la mesura hereterà la subclasse de circuit). A la configuració *serial_1_2*, els valors d'una mesura seran les posicions dels potenciòmetres de to i volum:

- X i Y. Son els valors dels potenciòmetres de volum i to, respectivament. Es guarden com a *float*.

A part d'aquests valors, també es guarda una clau que s'instancia automàticament quan creem una nova mesura. L'identificador de la clau és un enter auto-generat. Aquest valor no està especificat al model, doncs una entitat, per definició d'*NDB* té una clau.

Circuit. Representa un circuit creat a partir de pastilles i potenciòmetres. Es tracta d'una superclasse que és comú per totes les configuracions de circuits:

- *Pickup*. Es tracta d'un *String* que identificarà la pastilla del circuit. Tot i que hi haurà circuits amb més d'una pastilla, s'ha considerat imprescindible que un circuit posseeixi una pastilla. En casos amb més pastilles es podria considerar *Pickup* com la pastilla principal.
- *Measured*. És un booleà que ens indica si la pastilla del circuit ha de ser mesurada per SAN o no (*to-be-measured*). Aquest valor s'hereta de la pastilla un cop creem el circuit.

Igual que amb les mesures, es guarda una clau auto-instanciada al crear una nova entitat *Circuit*. L'identificador de la clau és un nom donat per l'usuari.

Dins les subclasses de *Circuit* tenim la configuració *serial_1_2* que és la configuració de circuit en sèrie amb una pastilla i dos potenciòmetres. Els seus camps son:

- *Volume*, un *String* que identifica el potenciòmetre de volum.
- *Tone*, un *String* que identifica el potenciòmetre de to.
- *Cc*. Es tracta d'un *Float* que indica el valor de la capacitat del cable.

Pastilla (*Pickup*). Representa una pastilla per guitarra elèctrica.

- *Rp*. Resistència de la pastilla en **sèrie** (*Float*).
- *Rpp*. Resistència de la pastilla en **paral·lel** (*Float*).

- *Lp*. Inductància de la pastilla (*Float*).
- *Cp*. Capacitat de la pastilla (*Float*).
- *Vol*, Volum de la pastilla (*Float*).
- *link_img*. Enllaç cap a la imatge que la representa (*String*).
- *Measured*. És un booleà que indica si la pastilla ha de ser mesurada (*to-be-measured*), és a dir, té valors *Rp*, *Lp* i *Cp* buits. Un cop mesurada, el valor de *Measured* és *False*.

La clau està auto-instanciada al crear una nova entitat i l'identificador és un nom donat per l'usuari.

Potenciòmetre (*Potentiometer*). Representa un potenciòmetre de volum o to.

- *R*. Representa la resistència del potenciòmetre.
- *Tone*. És un booleà i indica si es tracta d'un potenciòmetre de to (*True*) o de volum (*False*).
- *link_img*. Enllaç cap a la imatge que el representa (*String*).
- *Ct*. Representa, a partir d'un *Float*, la capacitat del condensador de to. Aquest valor només és vàlid quan el potenciòmetre és de to.

NDB també ens proporciona l'opció d'indexar o no les propietats de les entitats. Quan una propietat està indexada, significa que és accessible quan realitzem les consultes, és a dir, podem referir-nos a aquella propietat al cridar una consulta. Per exemple, els valors estàtics d'una pastilla no estaran indexats, doncs només els hem de guardar. Per altra banda, ens interessa poder separar, en una consulta, entre les pastilles que s'han de mesurar i les que no, pel que la propietat *measured* estarà indexada. Si evitem indexar certs valors, això ens permetrà més espai per dades que, pel contrari, estaria destinat als índexs.
Òbviament, les claus de les entitats estan sempre indexades.

Consultes

Les consultes que fem sobre *NDB* son molt senzilles. *NDB* ens proporciona funcions molt senzilles per realitzar-les. Cada model té la seva funció *query*, que permet seleccionar totes les entitats d'aquell mateix model, així com l'opció d'afegir-hi condicions. Si tinguéssim models amb claus amb diferents pares, tenim l'opció d'especificar que es busquin entitats d'un model Model amb el pare 'p' de la següent manera:

```
Model.query(ancestor=p)
```

També podem consultar entitats individuals si usem la funció *get_by_id*, indicant l'id proporcionat.

Per fer les insercions, en tenim prou amb instanciar una entitat del tipus de volem i usar la funció del model *.put()*. S'ha de tenir molt en compte que, a diferència d'altres bases de dades, *NDB* sobreescriu les entitats. Això significa que si creem dues instàncies d'un mateix model amb el mateix *id*, una sobreescriurà l'altra. Per tant, .put() funciona tant per crear com per actualitzar.

Per últim, si volem eliminar una entitat, el que NDB ens permet és eliminar la seva clau (`Model.key.delete()`). D'aquesta manera, deixem la informació d'aquella entitat inaccessible.

2.7. Sistema d'informes (SREP)

2.7.1. Introducció

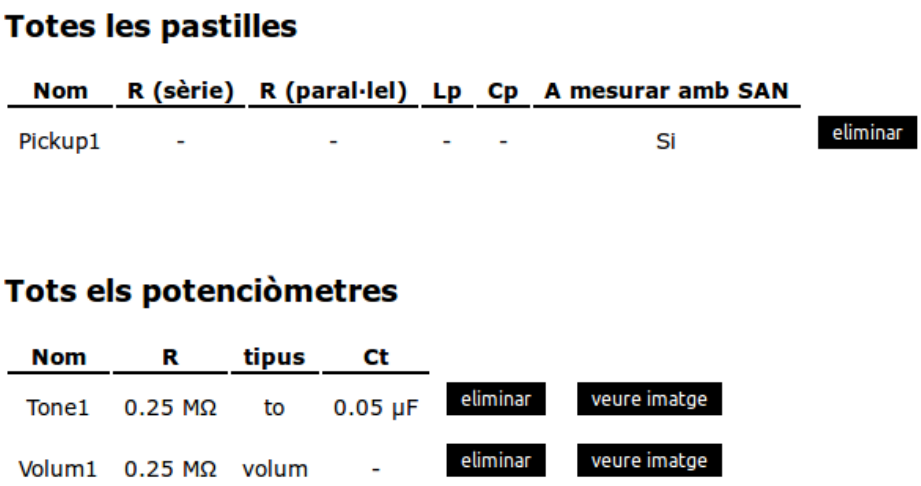
Arribem a l'últim subsistema a descriure. SREP respon a les necessitats de representar la informació obtinguda a partir de SED i SAN i emmagatzemada per SREC. El què ens ofereix SREP és la visualització de la informació de diferents maneres, segons ens convingui. També s'inclou la gestió d'aquesta informació.

Per implementar-lo, no hi ha manera més senzilla que fer-ho sobre la mateixa plataforma on s'hi troben SED, SREC i part de SAN. És a dir, l'aplicació web proveïda per *Google App Engine*.

2.7.2. Visualització

SREP té la funció principal de representar informació arribada des del altres subsistemes. En el fons tota aquesta informació li arriba a través d'SREC.

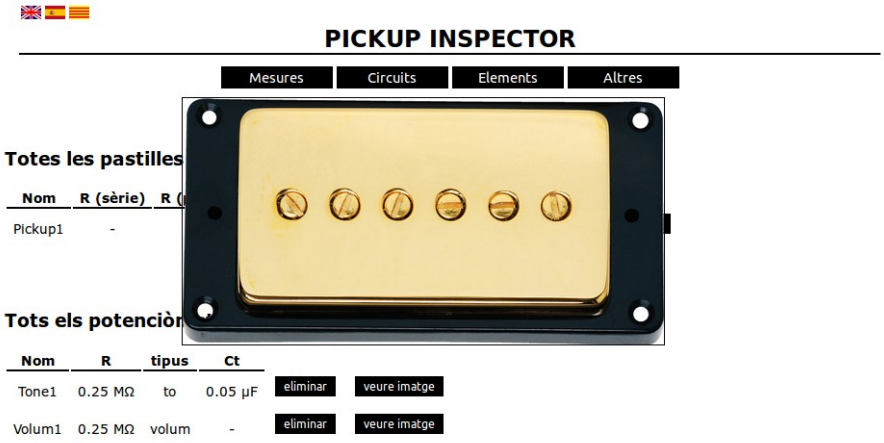
Un cop hem creat un element amb SAN o SED, ja sigui una pastilla o un potenciòmetre, podem veure els resultats d'aquesta acció fent la petició */element/list*.



Imatge 2.31. */elements/list* ens torna les llistes dels elements disponibles

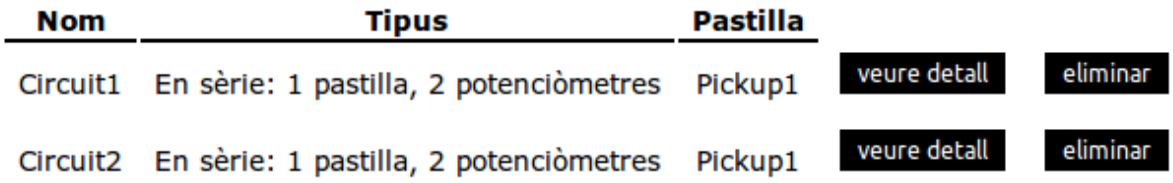
Una pàgina web ens proporciona un parell de taules, una per pastilles i l'altra, per potenciòmetres. A la taula de pastilles hi trobarem els valors estàtics d'aquesta. És a dir, la resistència en sèrie, la resistència en paral·lel, la inductància i la capacitat, a més del seu nom. També se'ns mostrarà el nom que identifica la pastilla i el valor del booleà *measured* que ens indicarà si la pastilla ha estat mesurada (és a dir, els valors estàtics no son buits) o no. En cas que els valors estàtics no s'hagin mesurat, la taula així ens ho indicarà i els valors a la taula simplement estaran buits.

Sota d'aquesta taula podem trobar la taula de potenciòmetres. De la mateixa, se'ns mostraran el valor de la seva resistència i, en cas de ser un potenciòmetre de to, la capacitat del seu condensador. Per tant, dins la taula hi trobarem una columna que indica el tipus de potenciòmetre que consultem i el nom amb el qual s'identifica.



Imatge 2.32. Visualització de la imatge d'una pastilla al prémer *see image*

Quan creem un circuit, */circuit/list* ens mostrarà una simple taula amb els noms dels circuits. Per tenir una referència més clara sobre el circuit que tenim, hem decidit incloure el l'identificador de la pastilla, a més de la configuració del circuit. Si volem accedir a tota la informació d'aquest, podem fer-ho amb el botó que ens apareixerà al costat anomenat *see detail* (veure detall). Aquí, se'ns mostrarà totes les dades del circuit en el moment de crear-lo, a més de la referència visual que ja havíem vist a SED. Així, tindrem els valors de la pastilla, el potenciòmetre de to i el potenciòmetre de volum que hem utilitzat i, el valor de la capacitat del cable si es que l'hem fet servir.



Imatge 2.33. Llistat de circuits

Hem decidit crear la pàgina de detall del circuit perquè en una taula era complicat representar i visualitzar tota la informació relativa al circuit. Però d'altra banda, necessitàvem poder visualitzar la llista de circuits que tenim al nostre sistema.

Hem inclòs la taula de mesures fetes sobre un circuit a la pàgina en detall, donat a que ens dona informació addicional sobre el circuit i ens permet gestionar i visualitzar mesures semblants amb més senzillesa que si ens trobem, per exemple, al llistat total de mesures.

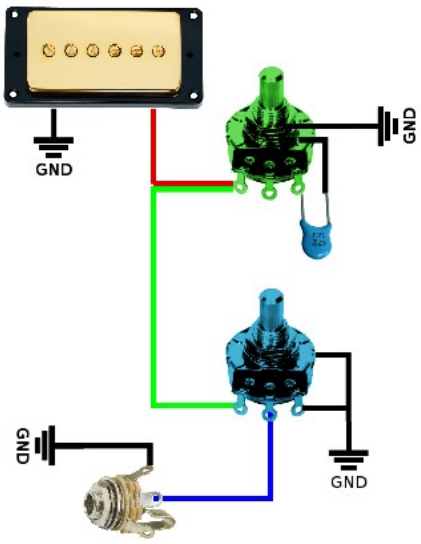
Eliminar

Nom: Circuit1
Tipus: En sèrie: 1 pastilla, 2 potenciòmetres

Pastilla
Nom: Pickup1
Resistència en sèrie: 8.82 kΩ
Resistència en paral·lel: 0.25 MΩ
Inductància: 4.72 H
Capacitat: 1.01 nF
A mesurar amb SAN: No

Potenciòmetre de to
Nom: Tone1
Resistència de to: 0.25 MΩ
Capacitat de to: 0.05 µF

Potenciòmetre de volum
Nom: Volum1
Resistència de volum: 0.25 MΩ



Mesures d'aquest circuit

ID	Data	F	Q		
4785074604081152	2014-11-21 16:15:48.275855	1.498 kHz	0.0392643318351	veure detall	Eliminar
6192449487634432	2014-11-21 16:15:43.934822	3.707 kHz	0.0463386703124	veure detall	Eliminar
5066549580791808	2014-11-21 16:15:38.456120	3.707 kHz	0.463386703124	veure detall	Eliminar
5629499534213120	2014-11-21 16:09:24.695222	3.516 kHz	0.222546611656	veure detall	Eliminar

Imatge 2.34. Detall del circuit amb el llistat de les seves mesures

Quan es realitza una mesura amb l'aplicació web de SAN, és a dir, es mesuren els valors dinàmics, aquesta es guarda a SREC. Podem veure una llista de les mesures realitzades si accedim a */measure/list*. A aquesta taula hi trobarem l'id de la mesura, el nom del circuit sobre el que s'ha fet la mesura, el nom de la pastilla del circuit, els seus valors dinàmics i la *timestamp*. Creiem útil tenir la *timestamp* a la llista doncs tenim les mesures ordenades segons aquesta variable. A la vegada, podem identificar més fàcilment una mesura realitzada en un determinat moment que una mesura feta sobre el nom d'un circuit o els seus resultats.

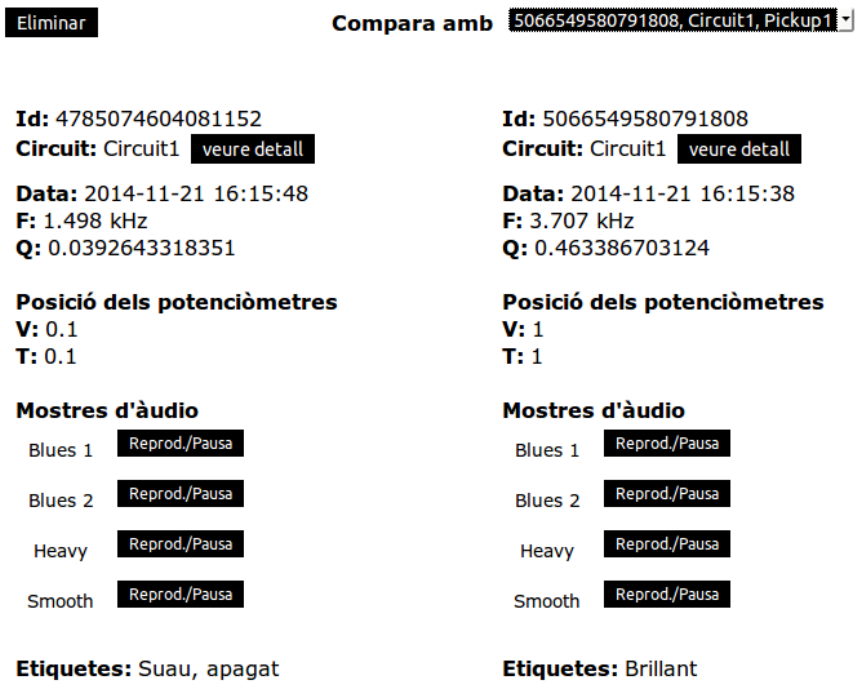
ID	Circuit	Pastilla	Data	F	Q		
4785074604081152	Circuit1	Pickup1	2014-11-21 16:15:48	1.498 kHz	0.039	veure detall	eliminar
6192449487634432	Circuit1	Pickup1	2014-11-21 16:15:43	3.707 kHz	0.046	veure detall	eliminar
5066549580791808	Circuit1	Pickup1	2014-11-21 16:15:38	3.707 kHz	0.463	veure detall	eliminar
5629499534213120	Circuit1	Pickup1	2014-11-21 16:09:24	3.516 kHz	0.222	veure detall	eliminar

Imatge 2.35. Llistat de mesures

Com amb les altres llistes, tenim un botó per cada mesura que ens permet accedir a la pàgina en detall. En el cas de les mesures, resulta molt més necessari l'ús d'una pàgina en detall, doncs la informació extreta és molt major.

Per una banda, accedim a la informació bàsica de la pastilla: els valors dinàmics, el nom del circuit (amb l'opció d'accedir al seva pròpia pàgina de detall), la data (*timestamp*), els valors de les posicions dels potenciòmetres i l'id de la mesura. Hem cregut interessant incloure l'identificador de la mesura, tot i que es tracta d'un valor auto-generat.

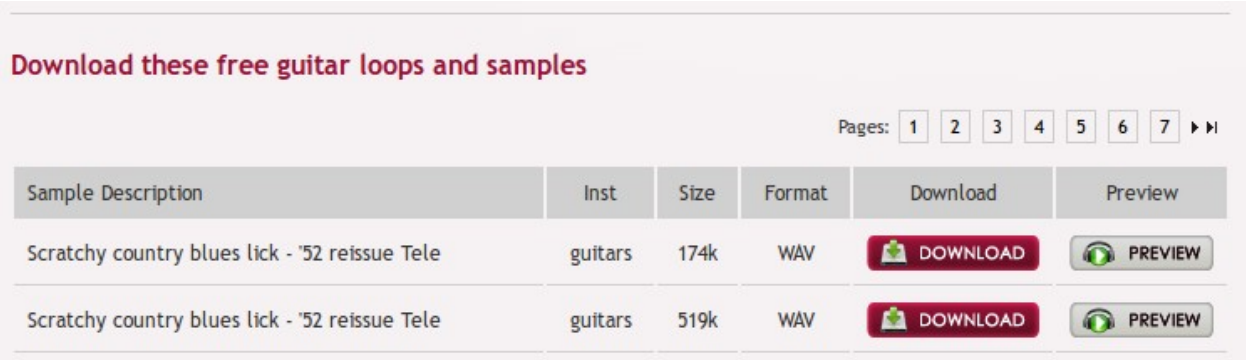
No estàvem disposats a generar els nostres propis id's, doncs resultava una feina massa farragosa pel servei que donaria i, després d'afegir noms a les pastilles, potenciòmetres i circuits, afegir noms a les mesures semblava quasi ridícul i molt abstracte.



Imatge 2.36. Detall d'una mesura amb comparació

Hem cregut necessari afegir l'identificador a l'hora de generar una llista de mesures per tal de ser comparades. Així, quan l'usuari accedeix a la pàgina de detall, tenim un desplegable on triar l'id d'una altra mesura i poder comparar els valors d'ambdues mesures. També hem cregut que podria servir de bona referència afegir-hi el nom del circuit i el nom de la pastilla.

A part d'aquesta informació, que no deixa de ser informació que ja hem pogut veure a SED i SAN, la vista en detall de la mesura ens proporciona molta més informació. Per una banda tenim les mostres d'àudio (*Audio Samples*). A partir d'uns arxius d'àudio de guitarra i el filtre proporcionat per *Web Audio API*, podem conèixer millor el so que ens proporciona el nostre circuit. Els àudios utilitzats son mostres d'àudio gratuïtes extretes de la pàgina web *freemusicloops.co.uk*. Als àudios se'ls hi aplica un filtre basat en els valors dinàmics de la mesura.



Imatge 2.37. Cerca d'arxius a *FreeMusicLoops.co.uk*

El resultat final no es pot considerar una prova precisa de com sonarà la nostra guitarra: primer de tot, perquè, per tenir un valor real del filtre, faria falta tenir el so d'una guitarra sense cap filtre, és a dir, sense cap circuit. Per aconseguir-lo, ens faria falta obtenir el so d'una guitarra el més net possible. Donat a que el so d'aquesta guitarra ja estaria condicionat per un filtre, faria falta 'treure-li' aquest filtre o, el que és el mateix, aplicar-li un filtre invertit. A partir d'aquesta base 'neta' li podríem aplicar el filtre de la mesura que estem tractant per tal d'obtenir el resultat desitjat.

A part d'això, el so generat pel reproductor del navegador més el so generat per l'equip de so que tingui el nostre PC, distorsionaria el so que en pogués sortir. Considerem que aconseguir aplicar-li un filtre invers a un àudio és treball suficient com per desenvolupar-ho amb més precisió en un altre projecte.

Tot i això, hem decidit incloure igualment aquestes mostres donat a que el nostre objectiu final és donar nous mecanismes a l'usuari perquè pugui valorar més objectivament el so de la seva guitarra. Tot i que les mostres d'àudio no siguin precises, poden proporcionar informació addicional que pot ser de molta més utilitat per l'usuari que altres dades molt més precises. Hem decidit incloure 4 pistes d'àudio pensant en diferents estils que poden ser fàcilment associats a gustos d'una gran gama d'usuaris. Així doncs hem inclòs dues mostres de **Blues**, donat a que inclou els sons de Rock i música negra; hem inclòs una mostra de guitarra distorsionada tocant **Heavy Metal**, un estil molt popular i que molts usuaris s'hi poden sentir identificats i, finalment, hem inclòs una mostra més calmada per tenir una referència més allunyada dels altres sons.

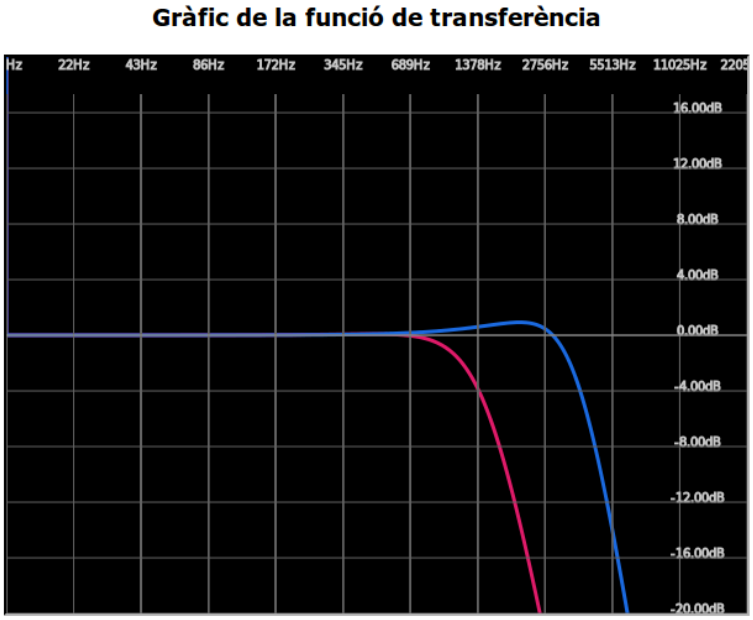
Les mostres també resulten una bona referència quan estem comparant dues mesures. Com ja hem dit, podem accedir a la informació d'altres mesures si, a partir del desplegable de la pàgina de detall, seleccionem un identificador. Immediatament se'ns carregaran les dades referents a la mesura amb aquell identificador. Se'ns inclourà la mateixa informació que tenim per la mesura principal. Així doncs, podem visualitzar la informació de les mesures, veure les configuracions d'una i altra i veure els diferències entre els valors emesos.

Aquesta informació pot ser especialment interessant si treballem amb diverses mesures amb petits canvis de configuració (per exemple, canviant les posicions dels potenciòmetres dins un mateix circuit). D'aquesta manera, veurem com evoluciona la freqüència de ressonància i la qualitat de les mesures respecte la mesura principal i, sobretot, podem veure el canvi de so que s'origina a partir de les mostres d'àudio modificades pels diferents filtres. És aquí on la mostra d'àudio guanya més força.

La pàgina en detall també ens mostra un gràfic que correspon a la corba creada per la funció de transferència sobre el rang de freqüències. Aquesta corba ens mostra un pic, on la **x** d'aquest punt representa la freqüència de ressonància i, la **y**, en representa la qualitat. Aquesta informació és la mateixa que ens proporcionava, fins ara, el sistema *Pickup Inspector*.

A diferència de la resta d'informació proporcionada al comparar una mesura, la corba creada per la mesura a comparar és dibuixada sobre el mateix graf amb un color diferent. La corba principal es dibuixa en un color magenta, mentre que la corba a comparar té un color blau. Hem cregut interessant fer aquesta aportació, doncs les dues corbes per separat eren difícils de contrastar, especialment si aquestes eren molt semblants. Aquesta informació pot ser menys interessant de cara a l'usuari, però serveix, un cop més, per aportar nous mètodes de valoració.

Per últim, la pàgina de detall de la mesura proporciona un últim camp, les etiquetes (*tags*). Aquests camps estan inspirats en la classificació de freqüències de ressonància que fa Helmuth Lemme a *Electric Guitar: Sound Secrets And Technology*. Les etiquetes venen a ser valors **sintestètics** (és a dir, valors propis d'altres sentits com el tacte o la vista) que descriuen el so del circuit. Fent una mica de recerca per fòrums d'Internet, veurem que molts d'aquests adjectius son usats normalment pels guitarristes i altres potencials usuaris. Decidim llavors, que son una bona referència i que poden ser mecanismes útils per valorar el so.



Imatge 2.38. Gràfic de la funció de transferència a la pàgina de detall d'una mesura

Les etiquetes son els adjectius que trobaràs a l'apartat de teoria del principi (*Com afecten els valors de sortida*).

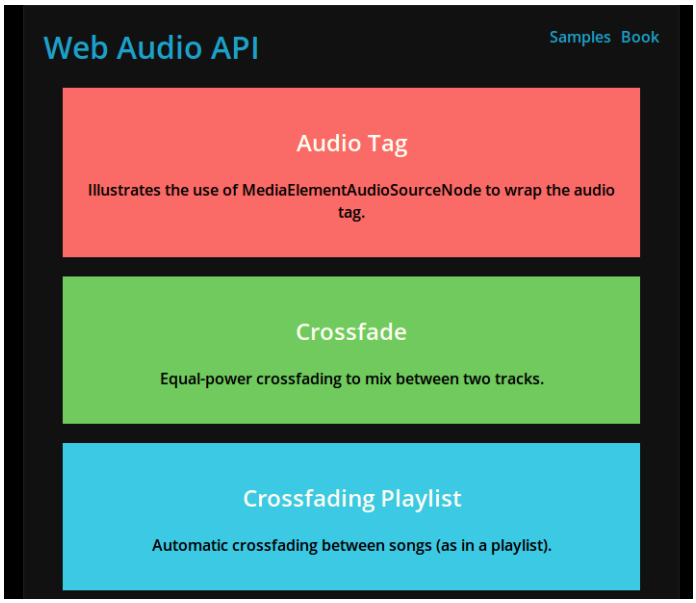
També s'ha inclòs la visualització de les unitats de mesura. Per cada valor reportat, tindrem la seva unitat de mesura que la representi. Si bé a SED donàvem l'opció d'especificar l'escala en la que volíem introduir les capacitats, per SREP hem decidit expressar els valors segons una petita funció. Aquesta calcularà l'escala de la unitat de mesura segons el valor que tinguem guardat. Les unitats de mesura tendeixen a la baixa i estan dividides cada 3 unitats: si tenim una resistència de 250000 Ohms, SREP ens mostrarà que la nostra resistència és de 0.25 MOhms (si tendís a la alta, tindríem 250 kOhms).

2.7.3. Web Audio API

Per realitzar les consultes gràfiques i les referències d'àudio, hem utilitzat *Web Audio API*. Es tracta d'una *API* programada amb *C++* i controlada a partir de *Javascript* que ens permet gestionar arxius d'àudio i modificar-los. L'*API* es basa en **nodes d'àudio** (*AudioNode*), elements interconnectats en una cadena que donen per resultat l'àudio renderitzat. Així, podem aplicar varis filtres, *fade-in*/*fade-out*'s, reverberació o altres efectes de manera senzilla i atomitzada.

A part d'això, *Web Audio API* permet reproduir so a molt baixa latència, la reproducció remota (és a dir, a altres dispositius o a través d'altres dispositius) amb *WebRTC* o l'ús d'una interfície d'anàlisi en temps real. Una de les grans avantatges d'*Audio Web API* és que funciona a través de *JavaScript* i *HTML5*, sense necessitat d'instal·lar extensions com *QuickTime* o *Flash*.

Per dibuixar la gràfica de la funció de transferència es va usar un codi d'exemple disponible a la pàgina oficial de *Web Audio API*. El codi, desenvolupat per **Chris Rogers** i modificat per **Chris Wilson**, ens permet dibuixar la corba de la funció de transferència a partir d'uns controladors que donen valor a la freqüència de ressonància i a la qualitat. És a partir d'aquesta implementació que modificarem el seu codi per adaptar-lo a les nostres necessitats.



Imatge 2.39. A *webaudioapi.com/samples* podem trobar diversos exemples de *Web Audio API*

En el nostre cas no fa falta disposar dels valors dels controladors, doncs el valor de la freqüència de ressonància i el valor de la qualitat son sempre els mateixos.

El que si que s'ha hagut de canviar ha estat l'escala de decibels, doncs el codi original arribava als 40db. En el nostre cas, donat a que el valor de la qualitat està normalitzada, no arribarà a penes als 2db.

El codi inicial semblava no funcionar al navegador Firefox, pel que s'ha modificat la inicialització del context d'àudio (*AudioContext*), que es tracta de la base per poder connectar-hi tots els nodes. N'hi ha hagut prou en canviar aquest codi:

```
<codi> context = new webkitAudioContext(); </codi>
```

per aquest:

```
<codi>if (typeof AudioContext !== "undefined") {
          context = new AudioContext();
} else if (typeof webkitAudioContext !== "undefined") {
     context = new webkitAudioContext();
} else {
     throw new Error('AudioContext not supported. :(');
}</codi>
```

Per tractar l'àudio, s'han fet servir arxius en format *.wav*, doncs tant *Firefox* com *Chrome* suporten aquest format i s'ha fet servir el node d'àudio BiquadFilterNode. Aquest node permet la creació de diferents tipus de filtre. Per el nostre cas hem usat un **filtre de pas baix** (*Low Pass Filter*). La interfície de *Web Audio API* ens permet la introducció directa dels valors *f* i *q* dins el filtre, pel que resulta molt senzill d'usar i modificar.

2.7.4 Gestió

Un altre dels apartat d'SREP és la gestió de la informació que rebem. Aquesta gestió inclou l'ordenació de les llistes i l'eliminació d'informació. S'ha considerat no tractar la modificació d'elements donat a una qüestió de manteniment de la informació: si modifiquem un circuit, quina relació queda amb les seves mesures? La nova informació de les mesures hauria de ser, o bé modificada, o bé eliminada, doncs no es correspondria amb els valors inicials. Hem decidit, per facilitat d'ús, no implementar aquesta funcionalitat.

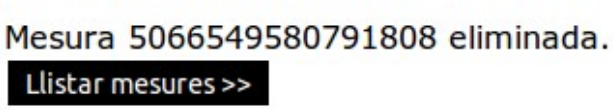
Tampoc s'ha tingut en compte la modificació de mesures doncs es tractaria d'un simple canvi dels valors de les posicions dels potenciòmetres. Això és, la creació d'una nova mesura, pel que ja existeix aquesta funcionalitat.

La modificació de pastilles o potenciòmetres podria ser possible, donat a que no estan enllaçades als circuits però hem pensat que, degut a la senzillesa de creació d'ambdós tipus d'elements, no feia falta aquesta funcionalitat.

Eliminació

Hem considerat que, donat a que no s'implementa la modificació de cap element, deixem oberta l'opció d'esborrat tots els elements. Per cada un dels elements tindrem un botó, ja sigui a llistes (circuits, mesures, pastilles i potenciòmetres) o a pàgines de detall (circuits, mesures) que ens permeti eliminar l'element desitjat.

Per evitar errors, sempre desplegarem una finestra alert de *Javascript* per confirmar l'eliminació de l'element triat. En el cas dels circuits, s'ha de recordar que l'eliminació d'un circuit implicarà l'eliminació de les seves mesures.



Imatge 2.40. Missatge d'èxit quan eliminem una mesura

En el cas dels potenciòmetres i pastilles i per evitar errors, es realitzarà una consulta sobre els circuits que continguin aquests components. En cas que hi hagi un o més circuits en aquesta consulta, no es permetrà eliminar la pastilla o potenciòmetre i així ens ho indicarà el programa.

Ordenació

Quan parlem d'ordenació de llistes, ens referim a la reordenació d'aquestes segons els diferents paràmetres que la conformen. Aquesta funcionalitat ha estat fàcilment implementada gràcies a la llibreria **sorttable.js** i la implementació de taules *HTML* a partir dels elements <*table*>, <*td*> i <*tr*>. El codi *sorttable.js* funciona automàticament si s'inclou a la nostre pàgina web i si, en una taula, li donem la classe *sorttable*. *Sorttable.js* associarà els valors de la primera fila de la taula com elements sobre els quals podem prémer i, automàticament, reordenar la taula seguint aquell paràmetre. Per exemple, les llistes de mesures poden ser reordenades segons el valor de la seva freqüència de ressonància si premem sobre *f* .

2.7.5. Plantilles

Per tal de realitzar una programació el més estructurada possible, s'han usat plantilles *HTML*. *Google App Engine* ens proporciona la llibreria *Jinja2*, orientada a la manipulació de fitxers. Aquesta mateixa llibreria és la que fem servir per obtenir les plantilles del sistema de fitxers. Els fitxers son enviats al client junt amb una sèrie de paràmetres que, gràcies a la sintaxi de *Jinja2* podem manipular i presentar com més ens convingui. Aquesta programació de les plantilles ens permet afegir poques línies *HTML* que poden ser iterades o condicionades segons ho reclami la pàgina. Per exemple, una pàgina on aparegui una llista, ens interessa poder escriure una sola línia que expressi la creació de la taula i programar el nombre de files segons el nombre d'elements que conté la llista a la base de dades.

Per implementar el patró *Layout*, hem usat una plantilla *serial_1_2.html* que generalitza la informació que s'hauria de presentar quan tractem un circuit amb d'aquesta configuració.

serial_1_2.html

```
<div id="paramsPickup"></div>
<div><span id="Rp_name">Serial resistance</span>: <span id="Rp"></span> <span id="RpF"></span>&#937;</div>
<div><span id="Rpp_name">Parallel resistance</span>: <span id="Rpp"></span> <span id="RppF"></span>&#937;</div>
<div><span id="Lp_name">Inductance</span>: <span id="Lp"></span> <span id="LpF"></span>H </div>
<div><span id="Cp_name">Capacitance</span>: <span id="Cp"></span> <span id="CpF"></span>F </div>
<div id="measured"></div>

</br>
<div id="paramsTone"></div>
<div><span id="Rt_name">Tone resistance</span>: <span id="Rt"></span> <span id="RtF"></span>&#937;</div>
<div><span id="Ct_name"> Tone capacitor</span>: <span id="Ct"></span> <span id="CtF"></span>F</div>
</br>
<div id="paramsVolume"></div>
<div><span id="Rv_name">Volume resistance</span>: <span id="Rv"></span> <span id="RvF"></span>&#937;</div>

</br>
<div><b><span id="others_title">Others</span></b></div>
<div id="paramsCable"></div>
</br>
```

Imatge 2.41. Plantilla de codi per *serial_1_2*

2.7.6. Estil i navegació

Una part menys funcional però necessària per una bona accessibilitat és la part de disseny. Aquesta part implica aplicar codi *CSS* a les pàgines mostrades, doncs la informació necessita ser estructurada per la seva fàcil visualització.

Hem mantingut un disseny senzill però elegant, utilitzant els colors blanc i negre, fent servir, com a font principal, *Verdana*. Hem separat el fons del contingut a partir d'un color gris, reduint el cos de la informació a un 80% de la pàgina i centrant-lo.

Diferències SAN

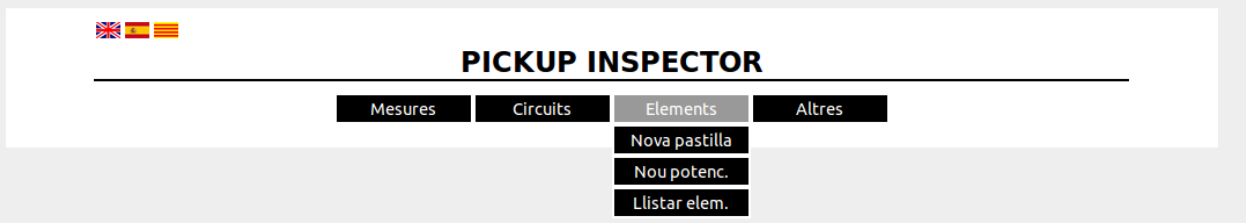
Aquest estil s'ha intentat aplicar de la mateixa manera a l'aplicació local SAN, per així, aconseguir una visió homogènia i global del nostre projecte. Malauradament, *Tkinter*, la llibreria amb la que està programada la versió gràfica de l'aplicació local no conté els mateixos mètodes que un navegador ni permet l'ús de *CSS* per estilitzar-la. En aquest cas, ens hem trobat amb el problema d'implementar la funcionalitat **hover**. Això és un *event* que es crida quan el nostre cursor es troba sobre un element. Quan aquest *event* passa, les propietat de l'element poden ser modificades. En el nostre cas només volem canviar el color de fons de l'element, passant de negre a un gris molt clar. Mentre que en *CSS* aquesta funcionalitat es fàcilment assignable a un element, amb *Tkinter* no s'ha aconseguit el mateix resultat. La funcionalitat *hover* si que funciona, però els colors obtinguts son els predeterminats pel sistema. Afortunadament, el fet d'aplicar un estil basat en negres, blancs i grisos, no ens ocasiona un gran canvi.

Capçalera

Per millorar la navegació entre les diferents pàgines, hem creat una capçalera que ens permet accedir a les diferents pàgines i subsistemes a l'aplicació web. La capçalera funciona com una plantilla *HTML* que es carrega a cada pàgina a partir de *Javascript*. Aquesta capçalera té diversos enllaços que ens porten a les diferents funcionalitats:

- *New Measure*. Ens porta a la pàgina de SED per crear una nova mesura.
- *All Measures*. Ens porta a la pàgina de SREP per visualitzar la totalitat de mesures.
- *New Circuits*. Pàgina de SED per crear un nou circuit.
- *All Circuits*. Visualització de tots els circuits.

- *New Pickup*. Creació de noves pastilles.
- *New Pot*. Creació de nous potenciòmetres.
- *All Elements*. Llistat de pastilles i potenciòmetres.
- *Documentation*. Pàgina per llegir la documentació i tutorial per l'usuari.
- *Source*. Aquí hi trobarem material per fer funcionar l'aplicació local de SAN i els seus requeriments.



Imatge 2.42. Capçalera desplegable

Per evitar un gran nombre d'enllaços, aquestes pestanyes han estat encapsulades en 4 blocs segons el seu origen.

- *New Measures* i *All Measures* estan a **Measures.**
- *New Circuits* i *All Circuits* estaran a **Circuits**.
- *New Pickup*, *New Pot.* i *All Ellements* a **Elements**.
- *Documentation* i *Source* a **Others**.

Aquestes pestanyes es despleguen amb els seus diferents enllaços quan ens hi posem a sobre.

2.7.7 Internacionalització

Per fer accessible el sistema a usuaris de diferents localitzacions, s'ha volgut internacionalitzar la visualització. S'han triat tres idiomes (català, castellà i català). Tant per les funcionalitats de la pàgina web com per l'aplicació local de SAN, s'ha utilitzat un mètode semblant de traducció de la informació.

Per evitar duplicar les pàgines, hem encapsulat cada paraula a ser traduïda dins un element d'*HTML* amb un identificador o classe concret. Per cada pàgina *HTML*, s'ha realitzat un arxiu *JSON* que conté, per cada identificador o classe que contingui un element a ser traduït, la seva versió en els tres idiomes, separats entre si. Llavors seguim el següent mètode: a partir d'unes banderetes presentades a la capçalera de navegació, triem quin dels idiomes volem fer servir. Aquesta acció modifica la *cookie lang* (*language*, en anglès) que guarda l'identificador de l'idioma. Llavors, un cop carreguem la pàgina, després d'haver carregat els elements *HTML*, es cridarà al servidor perquè ens serveixi l'arxiu *JSON* corresponent a la pàgina. Depenent de quin idioma haguem escollit, es carregarà un contingut o altre sobre els elements a ser traduïts.

D'aquesta manera, evitem repetir gran quantitat del codi i només canviar aquell contingut que necessita de traducció. D'altra banda *serialitzem* aquest contingut a partir dels paquets de *JSON*, que podem ser extensibles a altres idiomes.

```
"en" : {
    "title": "New Pickup",
    "pname": "Name",
    "resistance": "Resistance",
    "inductance": "Inductance",
    "capacitance": "Capacitance",
    "to_measure": "Measure with SAN",
    "drop_down": "Choose an image",
    "save": "Save Pickup"
},
"es": {
    "title": "Nueva pastilla",
    "pname": "Nombre",
    "resistance": "Resistencia",
    "inductance": "Inductancia",
    "capacitance": "Capacidad",
    "to_measure": "Medir con SAN",
    "drop_down": "Elige una imagen",
    "save": "Guardar pastilla"
},
"cat": {
    "title": "Nova pastilla",
    "pname": "Nom",
    "resistance": "Resistència",
    "inductance": "Inductància",
    "capacitance": "Capacitat",
    "to_measure": "Mesurar amb SAN",
    "drop_down": "Tria una imatge",
    "save": "Guardar pastilla"
}
```

Imatge 2.43. Plantilla d'internacionalització per la pàgina SED de creació d'una pastilla

2.8. Desenvolupament i evolució

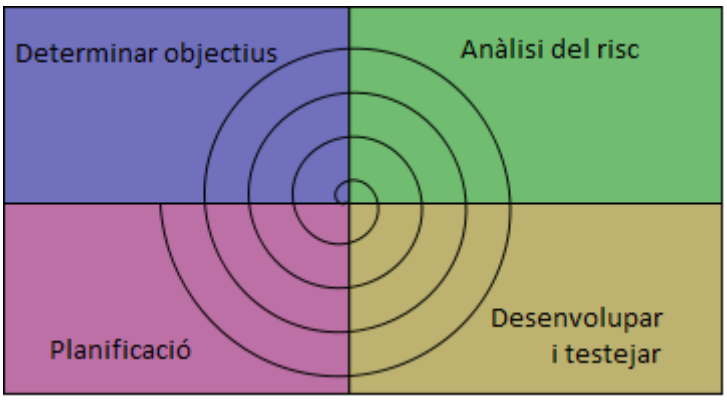
Com hem pogut veure el nostre sistema està composat de diversos subsistemes que, en el fons, s'acaben agrupant en dos grans blocs: l'aplicació web i l'aplicació local SAN, junt amb *Arduino*. És així com, al final, se li presenta a l'usuari. No obstant, ens ha semblat interessant representar les funcionalitats d'aquests blocs de manera separada per entendre la lògica que seguiríem i fer-ne un anàlisi exhaustiu.

El desenvolupament que s'ha seguit per generar aquests sistemes no ha estat separat, sinó necessàriament lligat entre si. És per això, que s'ha entès el sistema com una sola part a desenvolupar de manera conjunta. Per tal d'aconseguir-ho, s'ha realitzat un primer prototip, on s'hi representaven les necessitats més bàsiques del nostre sistema i, a partir d'aquí, s'ha seguit un model iteratiu per desenvolupar la resta de funcionalitats. A cada una d'aquestes iteracions l'hem considerat una nova versió del sistema.

Hem decidit usar aquest sistema per diverses raons. Primer de tot, perquè el desenvolupament en cascada està obsolet i l'intent de desenvolupar d'aquesta manera portaria a molts errors inesperats, així com canvis d'últim moment, llargues hores d'implementació i correcció i això repercutiria a poques hores de proves.

D'altra banda, el desenvolupament en iteració ens permet partir d'un prototip i, a partir d'aquest, desenvolupar diverses funcionalitats, mantenint així la consistència del producte i reduint el nombre d'hores dedicades a errors. Per tant, resulta un mètode més eficient. El fet de mantenir diverses versions (cada iteració) ens permet consultar una funcionalitat implementada amb anterioritat. A la vegada, el desenvolupador guanya experiència i bons hàbits a cada nova versió. Tenint en compte que *Google App Engine* és una eina nova a nosaltres, ens ha fet falta un aprenentatge que s'ha anat escalant a mesura que avançàvem en les versions.

De cara l'usuari, un desenvolupament en iteració permet distribuir una aplicació amb més antelació i anar-li oferint actualitzacions mentre es segueix desenvolupant.



Imatge 2.44. Representació en espiral d'una metodologia d'iteració

El model d'iteracions que hem seguit no segueix un model de desenvolupament concret (com podria ser *SCRUM*), però si que ha seguit unes normes generals:

- Un cop es comença una nova versió, es realitza un petit anàlisi de requisits i es descriuen les funcionalitats que hauria de tenir la versió.
- Aquestes funcionalitats son inamovibles excepte en casos excepcionals (mala integració al sistema, grau de complexitat inesperat, etc.). En cas d'un gran de complexitat inesperat, es divideix la funcionalitat en altres de més petites i s'implementa alguna d'aquestes i es deixen la resta per la següent iteració.
- Cada funcionalitat es desenvolupa independentment de les altres, excepte casos excepcionals (per exemple, realitzar la connexió entre dues parts). Les proves de funcionament de cada funcionalitat son també independents.
- Al acabar la llista de funcionalitats es fa una prova general del sistema de manera local i una altra un cop la versió està pujada al servidor. La versió ha ser correcta.

En el nostre cas, el desenvolupament s'ha fet en coordinació amb el projecte d'en Javier Vallés, pel que alguns requisits han estat especificats pels dos projectants. Tot i així, la majoria de funcionalitats les ha especificat el mateix desenvolupador (és a dir, jo) o el professor del projecte, doncs no hi havia un client extern per fer-ho. Per veure els documents de les versions es pot anar a l'annex indicat.

En qüestions de temps, el *timing* que s'ha seguit ha sigut irregular. Per la primera versió es va anar treballant poc a poc a partir d'Abril per temes laborals, mentre que la versió 2 s'ha desenvolupat intensament durant el Juliol. La versió 3 s'ha fet durant les primeres setmanes d'Agost i la versió 4 s'ha implementat al Setembre i principis d'Octubre.

Per tal de mantenir les versions de l'aplicació web, *Google App Engine* ens permet donar-li un número a cada una de les versions. D'aquesta manera, quan les pugem, aquestes versions queden separades entre si. Així podem accedir al codi de les anteriors versions i veure el seu funcionament. Una cosa que no s'ha aplicat és el control de versions de la base de dades. Així doncs, si creem elements amb la versió 1, aquests podran ser consultables amb la versió 2, 3 o 4. Això es tracta d'un error d'últim moment que no s'ha tingut en compte en el *testing*, doncs aquest es realitza independentment de les versions. Assumim que l'usuari només tindrà accés a una de les versions, pel que és un error menor.

	Version	Default	Deployed	Delete
○	1 instances | 744.97 KBytes | python27	No	135 days, 1:35:09 ago by undelluable@gmail.com	Delete
○	2 instances | 3.21 MBytes | python27	No	112 days, 3:11:22 ago by undelluable@gmail.com	Delete
●	3 instances | 3.32 MBytes | python27	Yes	74 days, 21:20:29 ago by undelluable@gmail.com	*Cannot delete default version.*
Make Default				

Imatge 2.45. Control de versions de *Google App Engine*

Pel desenvolupament de l'aplicació local de SAN s'ha seguit el mateix esquema, incloent les funcionalitats d'aquest programa al document de cada versió. Malauradament, el codi de SAN no es pot actualitzar de la mateixa manera que ho fa el codi de l'aplicació web. A la tercera versió s'ha inclòs el codi de SAN a l'aplicació web per poder ser descarregat i utilitzat per l'usuari.

De cara a les proves, s'ha desenvolupat el codi de manera local i, fins que no s'ha acabat una versió, aquest no ha estat pujat i provat al servidor. *Google App Engine* ens proporciona un sistema de fitxers i un servidor local per realitzar les nostres proves i mantenir els nostres arxius organitzats. Aquest ha estat de gran utilitat. Aquí s'ha de tenir en compte que, les proves realitzades amb l'aplicació local de SAN s'han de fet en comunicació amb el servidor local (*local-host*) mentre que al servidor de *Google App Engine* s'ha d'utilitzar una nova direcció.

3 Documentació i ajuda

3.1. Introducció

A part del sistema implementat, hem cregut necessari pel projecte, distribuir una documentació de cara a l'usuari final. Entenem que nosaltres, com enginyers informàtics, podem tenir certa habilitat en quan a abstracció de dades i en interpretació d'aquestes. Però això no ha de ser igual per l'usuari final. Considerem que entre usuaris finals, l'únic que tenen en comú és el fanatisme per les guitarres i la necessitat d'obtenir resultats que li puguin ser útils.

També s'ha de tenir en compte que es tracta d'un projecte prou original. El projecte parteix de la teoria d'un llibre i vol servir per trencar molts prejudicis que poden tenir els guitarristes avui en dia. El fet que no s'hagi distribuït més la teoria de Helmuth Lemme i no existeixin antecedents en quan a mètodes de mesura de pastilles i mètodes de valoració, fa que es creïn barreres entre el sistema i l'usuari. És per aquesta raó que s'ha de crear un pont entre els desenvolupadors / projectants i els usuaris, per tal que el que nosaltres hem desenvolupat, pugui ser entès pels usuaris i, sobretot, que aquests vulguin usar el nostre sistema.

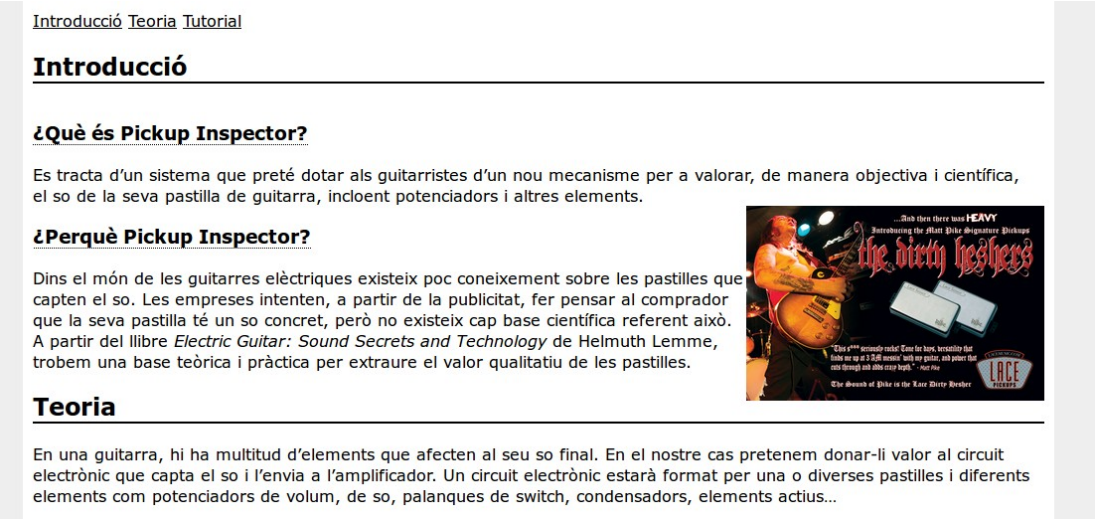
Ens farà falta, fer un resum de la teoria, perquè el projecte pugui tenir una base científica de cara als usuaris més escèptics i que sigui entenedor pels usuaris més novells. Per últim, es descriurà un tutorial per aprendre a utilitzar mínimament el sistema.

3.2. Teoria

La documentació de la teoria s'ha basat en el llibre de Helmuth Lemme, *Electric Guitar: Sound Secret & Technology*. Donat a que nosaltres també som novells en teoria electrònica, ens ha fet falta l'ajuda i consells d'en Javier Vallès. Dins la teoria, hi trobem tres apartats ben diferenciats: per una banda la introducció. Aquí expliquem què és exactament què és *Pickup Inspector* i a quines necessitats respon.

En segon lloc, descrivim, en termes molt resumits, quins son els valors a tenir en compte i com els podem mesurar. Aquí descrivim els valors d'entrada (valors de pastilla i potenciòmetres) i els valors de sortida (freqüència de ressonància i qualitat).

Per últim, la teoria ens presenta la classificació de les freqüències de ressonància que ja hem explicat anteriorment.



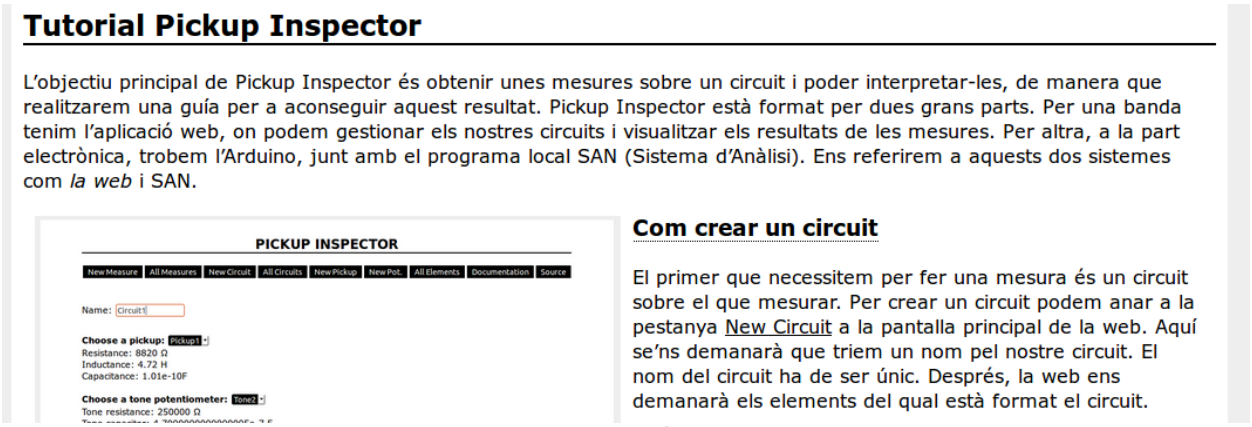
Imatge 3.1. Pàgina de teoria

3.3. Tutorial

El tutorial està pensat per ensenyar, pas a pas, el procés sencer del projecte. És a dir, des de que creem el primer potenciòmetre fins que comparem dues mesures. S'explica:

- com crear un nou potenciòmetre.
- com crear una pastilla, tant amb SED com amb SAN.
- com instal·lar el software requisit per fer funcionar SAN.
- com crear un circuit a partir de pastilles i potenciòmetres.
- com fer una mesura sobre un circuit.
- com interpretar les dades.
- com comparar dues mesures.

Amb algunes imatges i llenguatge senzill, creiem que faria falta la lectura i primera experiència d'un usuari principiant per millorar el tutorial.



Imatge 3.2. Visió del tutorial

3.4. Idiomes

Hem cregut convenient redactar la documentació en dos idiomes: català i anglès. Hem triat el català per ser idioma nadiu del desenvolupador i de fàcil comprensió pels usuaris més propers. L'anglès ha sigut idioma obligatori donat a que és l'idioma més internacionalitzat i, finalment, també hem inclòs el castellà, per ser la segona llengua més internacionalitzada. La documentació està estructurada sobre un *HTML* estàtic, pel que, per cada idioma, s'han hagut de convertir els caràcters especials amb l'eina web *htmlescape.net* i estructurar els paràgrafs i fotos, pel que, hem cregut que amb dos idiomes cobríem prou usuaris.

Una altra opció hagués sigut crear un *blog* a part (per exemple, a través de la plataforma *blogspot*) on s'hi disposés la documentació i es pogués editar sense necessitat de modificar un HTML. Així seria molt més senzill introduir noves traduccions. Només faria falta enllaçar aquest *blog* a l'aplicació.

Imatge 3.3. Les imatges de les banderes que hem fet servir

3.5. Altres funcionalitats

A part de les pàgines estàtiques de documentació, hem inclòs unes petites funcionalitats a aquesta:

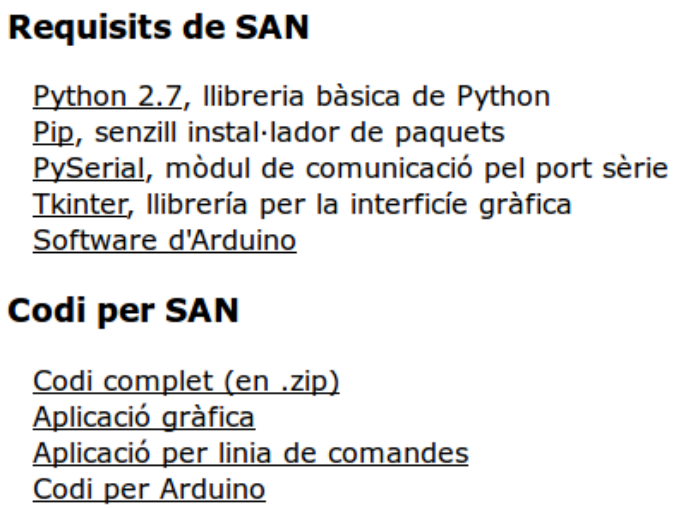
A partir d'un parell d'imatges, podem passar d'un idioma a l'altre. Les imatges estan dins un formulari que fa la petició de la pàgina directament al servidor. Segons la definició d'aquest tipus d'element (*input[type=image]*), el resultat de la petició ens retorna la pàgina, però també els valors de les coordenades del cursor un cop fem clic a la imatge. Sembla que no existeix un mètode senzill per evitar que això passi si seguim fent servir el tipus d'*input* descrit, pel que hem decidit deixar-ho com està, doncs no afecta al funcionament de l'aplicació.

Hem inclòs enllaços dins la pàgina que ens porten als diferents apartats de la documentació, a més de les diferents pàgines de l'aplicació web. Creiem que això és útil per mantenir una relació entre el tutorial i l'aplicació.

3.6. Source

A part de la documentació, s'ha inclòs una última pàgina a l'aplicació web que conté el codi necessari per fer córrer l'aplicació global SAN. Aquest codi inclou:

- La versió 2.7 de *Python*.
- El programa de *Python PiP* per fer més fàcil la instal·lació d'altres fitxers.
- La llibreria *PySerial*, necessària per fer la comunicació amb *Arduino*.
- La llibreria *Tkinter*, necessària per d'interfície gràfica de l'aplicació local.
- El fitxer *.ino* que ens permet fer les mesures sobre *Arduino* i controlar-lo.
- Un fitxer *.zip* on s'inclouen tots els paquets anteriors.



Imatge 3.4. Pàgina de descàrrega del programari necessari

4 Planificació

4.1. Primers passos

El projecte no ha seguit una planificació molt clara fins que s'ha començat el desenvolupament d'aquest. En un bon principi, el projecte no anava encarat al descrit aquí. Durant el Juny de 2013 tenia pensat contactar amb la *ONG* de la FIB **TxT** (Tecnologia Per Tothom) que es dedica a fer projectes informàtics a països en vies de desenvolupament, així com promoure el compromís social i ambiental en termes tecnològics. Malauradament, al moment en què vaig decidir posar-me en contacte amb un dels seus responsables, ja no hi havia ofertes de PFC's.

Imatge 4.1. Pàgina web de *TxT*

Així doncs, va tocar espavilar-se per trobar una oferta de projecte o desenvolupar una idea pròpia. Mentre es buscaven les ofertes, vaig contactar amb en **David López**, professor de la FIB al departament d'Arquitectura de Computadors, al qui li vaig proposar fer un projecte amb idea pròpia. La primera idea que tenia al cap era realitzar un projecte orientat en termes musicals, doncs sempre ha sigut una de les meves passions i sabia que, en David també n'era un apassionat. Es va decidir partir de l'idea de fer el projecte dins la classificació musical a partir de les emocions i l'anàlisi d'àudio. Quan la investigació va començar, ens vam adonar que ja existia una aplicació que feia això.

El projecte *A Mood-Based Music Classification And Exploration System* d'*Owen Craigie* ja complia aquest objectiu, tot aportant una base teòrica basada en psicologia i les relacions entre la música i les emocions humanes i diversos models de classificació emocional, a més de la integració de diferents sistemes d'extracció de dades d'àudio.

Una de les nostres propostes hagués sigut, o bé millorar el sistema a partir de l'obtenció del codi i la migració d'un sistema local a un de web, o bé realitzar un anàlisi d'altres tipus de classificadors en conjunt amb el projecte realitzat per *Craigie*.

En David va decidir ampliar la direcció del projecte a altres professors que els hi pogués interessar. Entre ells s'hi trobava en **Marc Alier**. Degut a la falta de temps d'en David i, aprofitant que, en aquells moment, en Marc era professor meu a l'assignatura de *GPS* (Gestió de Projectes de Software), vaig decidir parlar amb ell personalment. En Marc ja havia llegit la proposta de projecte però li va semblar que aquella idea no ens portaria a cap objectiu clar i tot quedaria a l'aire. Com a opció, en Marc va proposar el desenvolupament del sistema que tenim aquí.

4.2. Teoria, recerca i aprenentatge

Per començar a treballar en aquest projecte, va ser necessari dedicar moltes hores a la investigació i recerca. Primer ens va ser necessari llegir-nos la teoria de Helmuth Lemme, comprendre-la del tot i resumir les idees. Va ser necessari entendre els mètodes de mesura que s'utilitzaven per obtenir els valors fonamentals de les pastilles i investigar sobre les diferents alternatives electròniques que teníem per aconseguir-ho. Com ja hem explicat anteriorment, vam decantar-nos en l'ús de la plataforma *Arduino*.

Aquesta decisió va implicar l'aprenentatge d'*Arduino*, doncs no teníem una base teòrica ni pràctica sobre aquesta plataforma. Això ens va portar a realitzar el curs d'*AESS* sobre Introducció a *Arduino* i la versió avançada d'aquest. El curs es va realitzar durant les dues primers setmanes de Febrer de 2014.

Abans que es realitzés el curs, els primers prototips de SED ja estaven realitzats i algun ja s'havia descartat.

4.3. Coordinació

A mitjans de Febrer, ens arriba la proposta de la professora **Eva Vidal** i l'estudiant **Javier Vallés** de col·laborar en el desenvolupament del projecte. Aquí ens entrava un valor afegit al projecte, que era tenir una base teòrica sobre electrònica que des d'un bon principi no teníem. No obstant, això volia dir que el projecte es redefinia dins uns nous paràmetres.

Aquí va començar una fase de coordinació entre en Javier Vallés, en **Miguel García** (nou tutor d'en Javier), en Marc i jo. Aquesta coordinació va començar amb diverses reunions per a estructurar els objectius i possibilitats del projecte. Aquí es van començar a plantejar solucions informàtiques a la teoria desenvolupada per en Javi. Degut a la falta de temps d'ambdós projectants i problemes de salut, la part de coordinació es va estendre des de mitjans de Febrer fins a mitjans d'Abril.

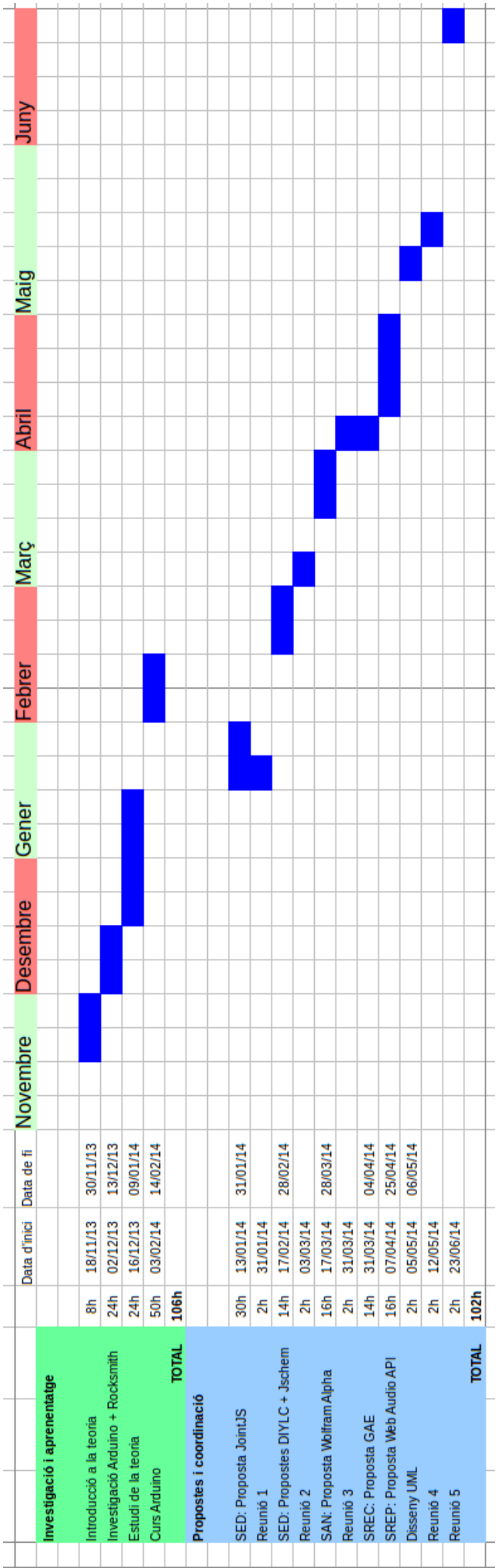
4.4. Implementació

A mitjans d'Abril ja comencem a desenvolupar realment l'aplicació real. Fins el final del projecte la coordinació segueix, però en un grau menor. Algunes parts teòriques no estaven clares, però n'hi han altres que comencen a agafar forma i donar resultats reals que ens seran vàlids pel sistema final.

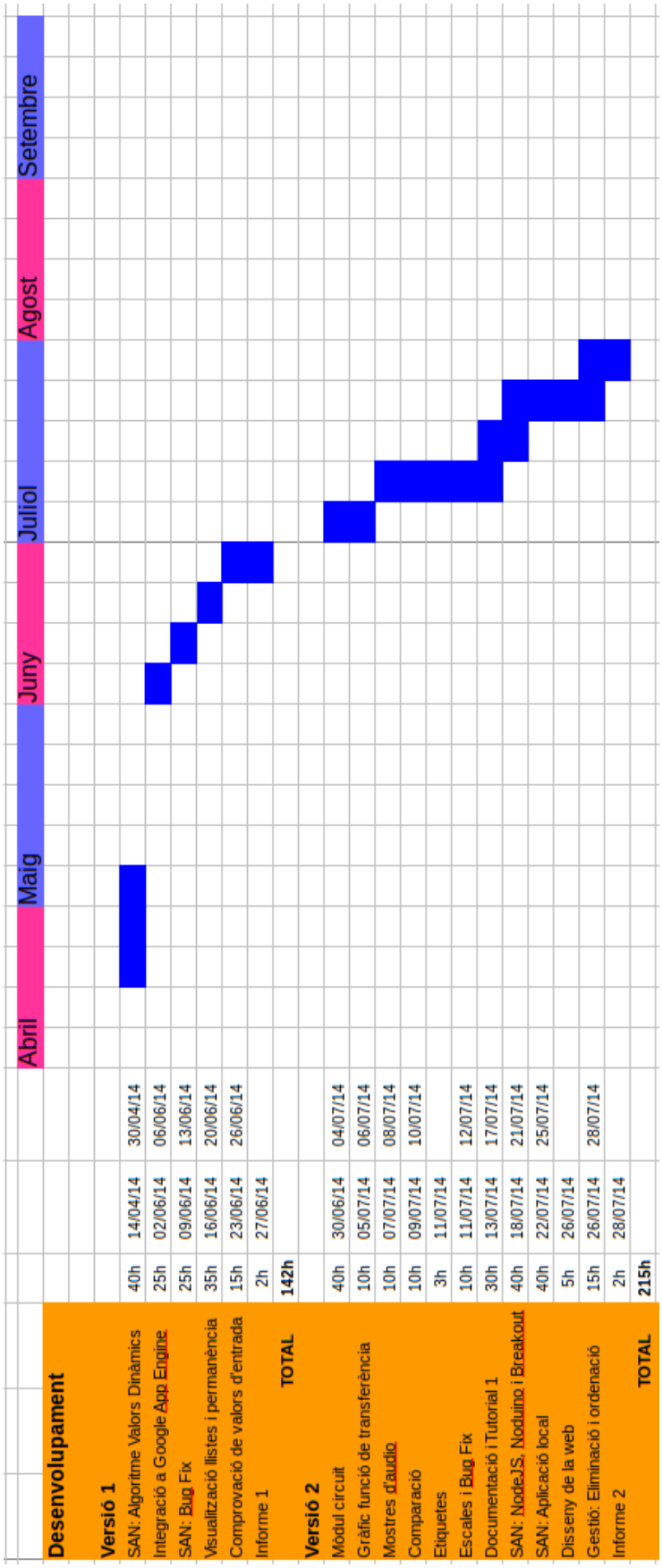
En aquesta etapa és quan decidim presentar el primer prototip i anar afegint noves versions sobre aquest. Aquesta part d'implementació comença a fins d'Abril i s'allarga fins l'Agost. En aquesta última etapa, durant el Juliol, es decideix redactar l'informe previ del projecte. Si tot va bé, presentarem el projecte a finals de Setembre o principis d'Octubre.

4.5. Mapa de planificació

Aquí es pot veure el mapa de planificació on s'hi mostren les hores realitzades. Per una millor visualització, l'hem separat en dos trams. La primera part inclou la investigació, aprenentatge, coordinació i presentació de propostes, mentre que la segona és la part de desenvolupament.



		Data d'inici	Data de fi
Investigació i aprenentatge			
Introducció a la teoria	8h	18/11/13	30/11/13
Investigació Arduino + Rocksmith	24h	02/12/13	13/12/13
Estudi de la teoria	24h	16/12/13	09/01/14
Curs Arduino	50h	03/02/14	14/02/14
TOTAL	**106h**		
Propostes i coordinació			
SED: Proposta JointJS	30h	13/01/14	31/01/14
Reunió 1	2h	31/01/14	
SED: Propostes DIYLC + Jschem	14h	17/02/14	28/02/14
Reunió 2	2h	03/03/14	
SAN: Proposta Wolfram Alpha	16h	17/03/14	28/03/14
Reunió 3	2h	31/03/14	
SREC: Proposta GAE	14h	31/03/14	04/04/14
SREP: Proposta Web Audio API	16h	07/04/14	25/04/14
Disseny UML	2h	05/05/14	06/05/14
Reunió 4	2h	12/05/14	
Reunió 5	2h	23/06/14	
TOTAL	**102h**		



Desenvolupament			
Versió 1			
SAN: Algoritme Valors Dinàmics	40h	14/04/14	30/04/14
Integració a Google App Engine	25h	02/06/14	06/06/14
SAN: Bug Fix	25h	09/06/14	13/06/14
Visualització llistes i permanència	35h	16/06/14	20/06/14
Comprovació de valors d'entrada	15h	23/06/14	26/06/14
Informe 1	2h	27/06/14	
TOTAL	**142h**		
Versió 2			
Mòdul circuit	40h	30/06/14	04/07/14
Gràfic funció de transferència	10h	05/07/14	06/07/14
Mostres d'audio	10h	07/07/14	08/07/14
Comparació	10h	09/07/14	10/07/14
Etiquetes	3h	11/07/14	
Escales i Bug Fix	10h	11/07/14	12/07/14
Documentació i Tutorial 1	30h	13/07/14	17/07/14
SAN: NodeJS, Noduino i Breakout	40h	18/07/14	21/07/14
SAN: Aplicació local	40h	22/07/14	25/07/14
Disseny de la web	5h	26/07/14	
Gestió: Eliminació i ordenació	15h	26/07/14	28/07/14
Informe 2	2h	28/07/14	
TOTAL	**215h**		

TOTAL	215h			
Versió 3				
Capçalera	5h	31/07/14	01/08/14	
Llista de mesures d'un circuit	10h	04/08/14	05/08/14	
Mòdul pastilles i potenciòmetres	25h	05/08/14	07/08/14	
Documentació: Traducció i menú	30h	07/08/14	09/08/14	
SAN: Bug Fix d'Aplicació Local	20h	11/08/14	14/08/14	
Disseny web i local	15h	15/08/14	17/08/14	
Informe 3	2h	18/08/14		
TOTAL	107h			
Versió 4				
Sliders	1h	01/09/14	01/09/14	
Escales	5h	02/09/14	02/09/14	
Imatges al crear	20h	03/09/14	05/09/14	
Construcció de l'imatge del circuit	20h	08/09/14	10/09/14	
Serial_1_2 Layout	10h	11/09/14	12/09/14	
Nova identificació de mesura	1h	15/09/14	15/09/14	
Internacionalització	40h	16/09/14	23/09/14	
Actualització SAN	30h	24/09/14	03/10/14	
TOTAL	127h			
TOTAL	591h			
Memòria i presentació	80h	19/08/14	15/10/14	
TOTAL				

5 Cost econòmic

El cost del projecte el dividirem entre el cost material dels components i el cost d'implementació d'aquest en termes d'hores treballades.

Al cost material li sumem les hores del curs d'*Arduino*. El servei d'allotjament de l'aplicació és gratuït, mentre que les peces del circuit de mesura de valors estàtics formen part del projecte d'en Javier.

	Preu
Arduino Starter Kit	100,00 €
Curs d'Arduino	120,00 €
TOTAL	220,00 €

Al cost d'implementació s'ha diferenciat la feina realitzar per un desenvolupador (Enginyer Informàtic) i un analista. El preu del desenvolupador és de **18038 €**[1] a l'any i el de l'analista és de **20672 €**[2]. Contant amb 2080 hores laborals, el preu per hora del desenvolupador son 8,7€ i el de l'analista, 9,9€.

	Hores	Preu	Total
Investigació i aprenentatge	106h	9,90 €	1.049,40 €
Coordinació i propostes	102h	9,90 €	1.009,80 €
Desenvolupament Versió 1	142h	8,70 €	1.235,40 €
Desenvolupament Versió 2	215h	8,70 €	1.870,50 €
Desenvolupament Versió 3	107h	8,70 €	930,90 €
Desenvolupament Versió 4	127h	8,70 €	1.104,90 €
Memoria	80h	9,90 €	792,00 €
TOTAL	879h		7.992,90 €

1 http://plandecarrera.infojobs.net/jobtitles.xhtml?jobtitle=analista
2 http://plandecarrera.infojobs.net/puesto-de-trabajo/ingeniero-informatico

6 Conclusions

6.1. Evolució del projecte

El desenvolupament del projecte ha estat molt irregular. S'han dedicat moltíssimes hores a la investigació i aprenentatge, així com la implementació de propostes que al final no han tirat endavant. Això ha estat degut a la falta de referents teòrics i pràctics que s'ha anat solucionant gràcies a la coordinació amb en Javier i l'aportació de noves propostes quan una altra quedava descartada. Si en podem treure alguna cosa positiva d'aquesta part, és que s'han descartat moltes propostes que no tenien sortida i hem reduït l'espai de treball.

En termes de temps, crec que el projecte s'ha allargat més de l'esperat degut a la redefinició del projecte en diverses parts i la necessitat de coordinació. També hi ha hagut problemes inesperats com problemes de salut i laborals. La coordinació ha sigut interessant de cara a tenir unes dates d'entrega de certes parts del projecte que ha permès aprofitar més el temps. També ha estat útil per tenir una molt bona base teòrica d'electrònica i sobre guitarres doncs en Javi en sap molt més que jo sobre aquests temes. A la vegada, poder-te comunicar amb un altre projectant implica poder resoldre dubtes i obtenir consells sobre temes no necessàriament centrats en el projecte, el que fa que el treball sigui amè. A la vegada però, alguns cops no hem deixat clar dates d'entrega, cosa que ha derivat a setmanes sense rebre material d'ambdues parts.

La part més complexa ha sigut la part d'implementació de les mesures d'*Arduino*, donat a que ha fet falta una coordinació entre en Javi i jo però, sobretot, degut al treball paral·lel amb la implementació l'aplicació web.

En la part d'implementació de l'aplicació web i l'aplicació local SAN és on m'he sentit més còmode, treballant autònomament i molt intensament. Això ha permès un desenvolupament molt més àgil. Malauradament, aquest procés s'ha realitzat durant l'etapa d'estiu i el *feedback* rebut tant per en Marc com per en Javi i en Miguel ha sigut baix.

De cara al codi, m'hagués agradat tenir més temps pel desenvolupament i poder deixar l'aplicació més preparada de cara a l'usuari final, agregant noves funcionalitats a partir de l'experiència d'usuaris externs.

6.2. Compliment dels objectius

De cara als objectius primerament plantejats, s'ha de dir que s'ha anat reduint l'abast a mesura que s'avançava amb el projecte. No ho veiem com un fracàs, doncs s'ha de tenir en compte que, quan es va començar el projecte no es tenia cap referència teòrica ni pràctica i aquesta s'ha hagut d'anar construint a mesura que s'avançava en el projecte, eliminant opcions no vàlides o massa complexes, fins a aconseguir unes guies que han pogut encaminar el projecte cap a un destí concret.

En relació als objectius descrits a l'informe previ d'entrega del projecte, podem dir que si que els hem complert. L'estudi sobre la teoria ens ha permès entendre quines eren les opcions i definir la implementació que volíem, a més de desestimar moltes propostes inviables. La implementació de les mesures estàtiques ha estat lenta però finalment s'han obtingut uns resultats correctes.

L'apartat d'implementació de l'aplicació local SAN i l'aplicació web també es pot dir que s'ha aconseguit, doncs el mètode iteratiu ha permès anar definint les diferents funcionalitats i arribar a bon port un cop s'acabava cada una de les iteracions de desenvolupament.

Hem aconseguit generar un mecanisme de mesura de pastilles i circuits elèctrics i implementar el sistema d'informació que permet la gestió d'aquestes dades. El guitarrista té una petita eina per valorar el so de la seva guitarra.

6.3. Propostes de millora

Com ja hem comentat, ens hagués agradat tenir més temps per treballar sobre l'aplicació web i l'aplicació local SAN. Si bé l'aplicació web funciona prou bé, creiem que té diversos errors:

SED. El sistema de disseny del circuit és molt bàsic. Aquest consta tan sols d'una interfície web sobre la que agreguem valors numèrics d'uns pocs elements. Hagués sigut interessant poder desenvolupar un sistema de disseny gràfic on s'hi representessin els elements i poder jugar amb un espai de 2 dimensions. També seria interessant poder agregar les nostres pròpies imatges per tal de fer el disseny més fidel a la realitat. Com ja s'ha comentat en aquest apartat, degut a limitacions d'algorítmia, aquesta part hauria d'anar lligada a l'augment de capacitat per descriure el circuit matemàticament i això implicaria un augment de la complexitat molt alt.

Una proposta de millora a curt termini seria la integració gràfica del circuit bàsic que tenim actualment. És a dir, es tractaria de desenvolupar una interfície de disseny gràfic que constés d'una pastilla, un potenciòmetre de to i un de volum amb algunes opcions de configuració. No es guanyarien noves funcionalitats però, tindríem la primera base per desenvolupar una interfície gràfica més completa.

Una altra alternativa podria ser incloure un camp de preu a cada element. D'aquesta manera, el sistema faria un càlcul del cost econòmic del nostre circuit. Així, els guitarristes podrien triar un component o altre segons un nou factor.

SAN. Aquest sistema compleix prou bé les seves funcionalitats. Per una banda, l'aplicació local milloraria amb l'ús d'una llibreria gràfica més completa i actualitzada que *Tkinter*. Una proposta seria migrar l'aplicació local a una interfície basada en *OpenGL*. També seria interessant donar-li funcionalitats extra com poder eliminar pastilles des d'allà o actualitzar-les en cas d'obtenir un error. De cara a la mesura dels valors estàtics de la pastilla, no creiem que es pugui millorar massa, doncs la seva funcionalitat està molt limitada.

En quan a comunicació entre SAN i la resta del sistema, la millora més notable seria l'autonomia de l'*Arduino*. Això implicaria noves tècniques de traspàs d'informació, ja fos a través d'una connexió *HTTP* amb connexió Wi-Fi (s'hauria d'incloure una placa *Xbee Explorer* a *Arduino*) o a través *Bluetooth* (s'hauria d'investigar aquesta opció). Aquesta dependència també voldria dir que hauríem d'estudiar l'autonomia energètica d'*Arduino*.

Per les mesures dinàmiques, ja hem comentat com aquestes queden molt limitades per la caracterització de la funció de transferència, pel que una inevitable millora seria desenvolupar una caracterització general per altres tipus de circuits. D'altra banda, el càlcul que es fa actualment està implementat amb *Javascript*. Aquest llenguatge està interpretat pel navegador, pel que no resulta molt eficient. Podríem migrar aquest codi a un llenguatge compilat com *C* o *C++*. Això implicaria integrar un programa *C* al servidor o migrar l'aplicació web a un programa local.

També és necessari dir que el fet d'implementar la nostra eina en codi obert ens ha limitat en l'ús de programari més potent com *Matlab*. Amb aquesta eina, les mesures, gràfiques i altres tècniques resulten molt més fàcils d'implementar.

SREC. En quan a la comunicació entre les parts, aquestes no necessiten de moltes millores. Hagués sigut interessant poder integrar l'aplicació local SAN i comunicar-nos amb l'*Arduino* a través d'un servidor. Si triéssim aquesta opció, faria falta o bé comunicar-nos a partir d'una connexió *HTTP* amb l'*Arduino* o migrar l'aplicació a una nova plataforma o servidor que si acceptés la llibreria *PySerial* de *Python*.

Tot i que estem molt satisfets amb la base de dades *NDB*, també seria desitjable generar un nou model de dades i lògica que impliqués menys redundància de dades. Això també ens permetria afegir més opcions als models. Per exemple, podríem guardar alguns valors amb la seva unitat de mesura, i poder-les proveir amb més informació.

SREP. Aquesta part seria la que més ens agradaria millorar. Primer de tot, seria del tot necessari implementar un sistema d'usuaris. Actualment el sistema està pujat al servidor de *Google App Engine* i té accés públic a tothom que el vulgui fer servir, crear nova informació o eliminar-la, pel que pot portar a males pràctiques fàcilment. Tot i això, es va decidir no implementar aquesta part per diverses raons.

Primer de tot, perquè el fet d'incloure un sistema d'usuaris no millorava cap funcionalitat que altres aplicacions no tinguessin. En segon lloc, la complexitat d'SREP creixia i s'havien de decidir la lògica d'aquest sistema. Els usuaris només podien veure les seves mesures o també les d'altres usuaris? I podien esborrar altres mesures? Feia falta agregar un administrador perquè pogués gestionar els usuaris? Quins drets tenia aquest administrador? Totes aquestes preguntes s'haurien de respondre en el cas d'afegir aquesta funcionalitat.

Això afecta negativament a l'escalabilitat del projecte. Si ens trobem en què diferents usuaris comencen a realitzar moltes mesures, o a generar molts elements, no sabem si la visualització d'aquests quedaria altament afectada. La nostra conta de *Google App Engine* està limitada en espai i, tot i que es descriu a sí mateix com un sistema escalable, no sabem quin seria el rendiment de l'aplicació.

El que també hagués sigut interessant hagués sigut generar un sistema de visites on quan una mesura rebés cert nombre de visites, passés automàticament a ser una mesura de referència o famosa. Això voldria dir que seria una visita destacada alhora de comparar les nostres mesures amb aquesta mesura famosa. També interessaria poder fer comparacions entre vàries mesures i no només amb dues.

De cara a les referències auditives, també faria falta millorar-les, tenint l'opció de pujar el nostre propi àudio o sent capaços d'aplicar l'idea de filtre invertit.

Com ja s'ha comentat, faria falta millorar el tutorial a través d'experiències d'usuaris novells i migrar la documentació a un *blog* amb traducció a altres idiomes.

6.4. Àmbits tractats

Dins el projecte hem pogut tocar diferents àmbits de la enginyeria informàtica:

- Per entendre la teoria i mesures, hem recordat conceptes de **física**, **matemàtiques** i **algorítmia**.
- Els algoritmes d'*Arduino* i la comunicació amb l'aplicació local SAN formen part de l'àmbit de **microcontroladors** i **comunicació entre sistemes**.
- La implementació de l'aplicació local està lligada a la **visualització** i **interfícies gràfiques**.
- Per implementar la web i el servidor hem treballat sobre els àmbits de **sistemes web** i **xarxes**.
- Per especificar i dissenyar els subsistemes hem fet ús de l'**enginyeria de software**.
- Per la persistència de dades, hem treballat l'apartat de **bases de dades**.
- Diverses funcionalitats, com Web Audio API, Arduino i Google App Engine formen part de l'àmbit de l'ús de **noves tecnologies de la informació**.
- Per últim, ens ha sigut molt útil l'estudi de **gestió de projectes** de cara a la gestió global del PFC.

6.5. Conclusions personals

6.5.1 Tecnologies

En aquest projecte he utilitzat tecnologies de tot tipus, des de totalment desconegudes fins a eines que ja teníem més per mà.

Per una banda, l'aprenentatge d'*Arduino* m'ha semblat molt valuós de cara a la realització de projectes personals. No sé fins a quin punt pot servir a nivell laboral. Si no hagués estat per triar aquesta plataforma, segurament no hagués fet el curs de formació d'*AESS* i m'hagués costat introduir-me a *Arduino*.

També ha sigut molt interessant aprendre l'ús de *Google App Engine* per les mateixes raons personals que amb *Arduino*. El que més he pogut apreciar de *Google App Engine* ha sigut l'ús d'un llenguatge amb el que em sento totalment còmode com és *Python*. M'ha permès desenvolupar sense grans problemes sobre un *framework* molt senzill i guanyar més experiència. També ha sigut molt interessant l'ús de la base de dades tipus *NDB*, donat a que es fa molt senzill la persistència de dades en contrast amb llenguatges del tipus *SQL*.

Tot i que la intenció ja era treballar sobre una aplicació web, aquesta decisió m'ha donat molta més experiència en aquest cap, que realment és un dels camps que més m'interessen en l'àmbit informàtic.

Tkinter ha sigut una eina de desenvolupament gràfic de *Python* molt senzilla d'utilitzar, però també molt limitada. Seria interessant practicar amb altres llibreries sobre el mateix llenguatge.

La connexió amb *Arduino* m'ha portat molts mals de caps degut a la poca experiència que teníem en aquest camp, portant-me a fer servir llenguatges o *frameworks* que no he acabat d'entendre o que simplement no han funcionat sobre el sistema operatiu que he fet servir. Així doncs, *NodeJS*, un *framework* t'han estès, no ha acabat d'agradar-me i se m'ha fet complicat. M'hagués agradat poder treballar realment amb altres *frameworks* de comunicació entre navegador i *Arduino* com *BreakoutJS* o *Noduino*, doncs la seva proposta em sembla molt interessant.

Tot i que *Javascript* sigui un llenguatge interpretat i ineficient, la quantitat de llibreries que se'n desprenen son enormes i ha faltat molt poca recerca per trobar eines molt interessants com *MathJS* o *JointJS*. Crec que *JointJS* té un gran potencial per a generar aplicacions online que no hem tingut l'oportunitat d'aprofundir en detall.

Web Audio API també és una gran eina que pot ajudar en la millora de la nostra aplicació, però també en altres aplicacions, inclús en ús comercial/laboral. Conté un enorme potencial en termes d'àudio i, tenint en compte el meu interès en temes musicals, aquesta *API* m'ha sorprès molt gratament.

Per últim, també hem de reconèixer l'experiència guanyada amb codi web (*HTML5*, *CSS*, *Javascript/jQuery*). Aquests llenguatges ja els coneixíem, però el projecte ens ha permès aprofundir en certes funcionalitats i a guanyar lleugeresa alhora de desenvolupar.

Teoria. Vam entrar en aquest projecte tenint molt poca idea sobre el funcionament d'una pastilla. Quan en sortim, ens sentim satisfets en els coneixements teòrics apresos. Tot i que a la pràctica aquesta no ens aporti molta utilitat (un no canvia de pastilla i circuit cada setmana), la teoria de Lemme i la documentació generada ens pot ajudar a millorar les nostres capacitats com a guitarristes.

6.5.2 Desenvolupament/Treball

El desenvolupament seguit durant el treball ens ha fet guanyar bons hàbits alhora de preparar-nos la feina, mantenir ritmes de treball i seguir mètodes de desenvolupament (en el nostre cas, en iteració). Esperem poder fer servir aquestes habilitats en entorns més grans.

En general es pot dir que s'ha après moltíssim en temes d'electrònica i en funcionament de pastilles, que s'ha après a comunicar-nos amb elements externs al dispositiu i que hem crescut en gran mesura en termes d'aplicacions web.

Actualment existeix molta informació a Internet que ens permet el desenvolupament d'eines molt útils sense necessitat d'entrar en detall en el seu ús. A aquestes eines se li ha de sumar la gran comunitat d'usuaris i desenvolupadors que es resolen els dubtes mútuament a través de pàgines com *stackoverflow*.

El món de les guitarres i les pastilles està massa poc explorat per la relació que té la humanitat amb la música.

Per últim, s'ha de dir que no considerem el nostre projecte com a tancat i animem a nous projectants a millorar-lo, doncs podria ser una gran eina pels guitarristes en poc temps.

7 Bibliografia

- Lemme, H. *Electric Guitar: Sound Secret And Technology.*
- *Arduino Starter Kit Tutorial*
- JEDI. *Python per a tots.*
- *Web Audio Api* [en línia]. Disponible a: webaudioapi.com
- *Arduino* [en línia]. Disponible a: arduino.cc
- *Python* [en línia]. Disponible a: www.python.org
- *Build Your Guitar* [en línia]. Disponible a: buildyourguitar.com
- *Rocksmith* [en línia]. Disponible a: rocksmith.ubi.com
- *AESS* [en línia]. Disponible a: aess.upc.es/moodle
- *SPICE* [en línia]. Disponible a: es.wikipedia.org/wiki/SPICE
- *Wolfram Alpha* [en línia]. Disponible a: www.wolframalpha.com
- *Node JS* [en línia]. Disponible a: nodejs.org
- *Socket IO* [en línia]. Disponible a: socket.io
- *Noduino* [en línia]. Disponible a: semu.github.io/noduino
- *BreakoutJS* [en línia]. Disponible a: breakout.js.com
- *Tkinter* [en línia]. Disponible a: wiki.python.org/moin/TkInter
- *PySerial* [en línia]. Disponible a: pyserial.sourceforge.net
- *stackoverflow* [en línia]. Disponible a: stackoverflow.com
- *Pip* [en línia]. Disponible a: pypi.python.org/pypi/pip
- *Iron Gear* [en línia]. Disponible a: irongear.co.uk
- *JointJS* [en línia]. Disponible a: jointjs.com
- *Jschem* [en línia]. Disponible a: dhost.info/jschem/
- *DIY Layout Creator* [en línia]. Disponible a: diy-fever.com/software/diylc/

- *jQuery* [en línia]. Disponible a: jquery.com
- *Javascript* [en línia]. Disponible a: www.w3schools.com/js/
- *HTML5* [en línia]. Disponible a: html5rocks.com/
- *CSS* [en línia]. Disponible a: www.w3schools.com/css/
- *AJAX* [en línia]. Disponible a: www.w3schools.com/ajax/
- *Google App Engine* [en línia]. Disponible a: appengine.google.com
- *Draw.io* [en línia]. Disponible a: www.draw.io
- *Free Music Loops* [en línia]. Disponible a: freemusicloops.co.uk
- *Sorttable* [en línia]. Disponible a: kryogenix.org/code/browser/sorttable/
- *Tecnologia Per Tothom (TxT)* [en línia]. Disponible a: txt.upc.edu
- Craigie, Owen. *A mood-based music classification and exploration system* [en línia]. Disponible a: http://dspace.mit.edu/handle/1721.1/39337
- *Math JS* [en línia]. Disponible a: mathjs.org
- *Can I Use* [en línia]. Disponible a: caniuse.com
- *Web Audio API – Getting started* [en línia]. Disponible a: creativejs.com/resources/web-audio-api-getting-started/
- *HTML Escape Tool* [en línia]. Disponible a: www.htmlescape.net/htmlescape_tool.html

Annex I: Glossari

Pastilla: transductor que capta, per inducció electromagnètica, les vibracions de les cordes metàl·liques als instruments musicals elèctrics. S'utilitza en instruments de corda com per exemple guitarra elèctrica, baix elèctric i guitarra hawaiana[3].

Potenciòmetre: component electrònic, un tipus de resistència variable que acostuma a utilitzar-se com a divisor de tensió. Hi ha una forma de potenciòmetre que s'utilitza com un instrument per mesurar la tensió elèctrica[4].

Condensador: dispositiu que emmagatzema energia en el camp elèctric que s'estableix entre un parell de conductors els quals estan carregats però amb càrregues elèctriques oposades[5].

Cable: un element conductor format per un conjunt variable de fils metàl·lics, generalment recoberts per un material aïllant o protector. Els conductors elèctrics són els cables que tenen com a finalitat el transport d'electricitat[6].

Circuit: conjunt simple o complex de conductors i components elèctrics o electrònics recorregut per un corrent elèctric[7].

Mesura: valor numèric o magnitud d'algun atribut físic d'un objecte, com per exemple la longitud, la capacitat, el volum o el pes. [...] El terme també s'utilitza per anomenar el procés per a arribar a obtenir el valor d'una determinada magnitud[8].

Arduino: plataforma electrònica de codi obert basat en maquinari i programari fàcil d'utilitzar. Està dirigit a qualsevol persona que fa projectes interactius.[9].

Google App Engine: és una tecnologia de la computació en núvol (cloud-computing) per allotjar aplicacions web en els centres de dades gestionats per Google[10].

Webapp2: *framework* web de Python, lleuger i compatible amb Google App Engine[11].

NDB: New Data Base. API que proporciona emmagatzematge i persistència de dades en models sense esuqema (schemaless)[12].

Port sèrie: és un tipus de maquinari de comunicació sèrie que permet la transmissió o la recepció de bits un darrere de l'altre[13].

3 ca.wikipedia.org/wiki/Pastilla_electromagnètica
4 ca.wikipedia.org/wiki/Potenciòmetre
5 ca.wikipedia.org/wiki/Condensador
6 ca.wikipedia.org/wiki/Cable_elèctric
7 ca.wikipedia.org/wiki/Circuit_elèctric
8 ca.wikipedia.org/wiki/Mesura
9 www.arduino.cc/
10 stackoverflow.com/questions/tagged/google-app-engine
11 webapp-improved.appspot.com/
12 cloud.google.com/appengine/docs/python/ndb/
13 ca.wikipedia.org/wiki/Port_s%C3%A8rie

Web Audio API: API dissenyada per manipular i reproduir elements d'àudio en una pàgina web o una aplicació [14].

Tkinter: paquet per defecte de Python per l'implementació de GUI's estàndard (Graphical User Interface). És una capa orientada a objectes basada en Tcl / Tk[15].

HTTP: protocol per a l'intercanvi de documents d'hipertext i multimèdia al web[16].

Funció de transferència: funció en el domini s que relaciona l'entrada i la sortida. La funció de transferència és igual al quocient de les transformades de Laplace de l'entrada E(s) i la sortida S(s)[17].

14 developer.mozilla.org/es/docs/Web_Audio_API
15 wiki.python.org/moin/TkInter
16 ca.wikipedia.org/wiki/Protocol_de_transfer%C3%A8ncia_d%27hipertext
17 ca.wikipedia.org/wiki/Funció_de_transferència